Analyse und Vorhersage des Flächen- und Energieverbrauches-optimaler Hardware Polynom-Multiplizierer für $GF(2^n)$ für elliptische Kurven Kryptographie

Von der Fakultät für Mathematik, Naturwissenschaften und Informatik
der Brandenburgischen Technischen Universität Cottbus

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
(Dr.-Ing.)

genehmigte Dissertation

vorgelegt von

Diplom-Radiophysikerin

Zoya Dyka

Geboren am 26.02.1972 in Kiew (Ukraine)

Gutachter: Prof. Dr. P. Langendörfer

Gutachter: Prof. Dr. K. Meer

Gutachter: Prof. Dr. H. Michalik

Tag der mündlichen Prüfung: 13. April 2012

Inhaltsverzeichnis

Extended Abstract

During recent years elliptic curve cryptography (ECC) has gained significant attention especially for devices with scarce resources such as wireless sensor nodes. Hardware implementations are considered to be the key enabler for using ECC on this class of devices. Out of the operations needed to execute ECC the polynomial multiplication is the one which is investigated most since it is one of the most complex field operations and executed very often.

The majority of research papers focuses on reducing the number of partial-multiplications while neglecting the increased effort for additions of the partial products. This thesis investigates how the latter can be optimized. A reduction of additions can be achieved by using pre-defined processing sequences for summing up partial products. In this work a method to find the optimized processing sequence is presented. It is applied to 10 multiplication methods of polynomials over $GF(2^n)$. For example when applied to the generalized Karatsuba multiplication [18] the optimized processing sequence saves up to 39 per cent of XOR-gates in average for polynomials with a length up to 600 bits.

In addition it is known that combining different multiplication methods reduced the total complexity of the multiplier. For example using the classical MM for calculation of small partial products in combination with other MMs can improve chip-parameters of the resulting multipliers. An optimal combination of several multiplication approaches for which the optimized processing sequence of XOR-operations is used reduces the area and energy consumption of the resulting multiplier significantly. This work presents an algorithm to determine the optimal combination of multiplication methods with pre-defined processing sequences for hardware implementation of an highly efficient polynomial multiplier in $GF(2^n)$. The combinations determined by this algorithm save in average 12 % of the chip-area for polynomials with a length up to 600 bits in comparison to data reconstructed from [30].

In addition the effect of the optimization techniques researched in this thesis was evaluated using the example of polynomial multiplier for 233 Bits long operands. The multiplier uses the Winograd-MM for segmentation of operands and executes partial multiplication using an optimized 78 bits partial multiplier. The theoretical results have been verified successfully by synthesizing this multiplier for the IHP 0.13 µm technology. In comparison to a synthesized version of the design given in [28] the optimized multiplier of this thesis reduces the energy consumption and execution time of the kP operation by 24 and 28 per cent, respectively.

Kurzfassung

Die Anwendung asymmetrischer Kryptosysteme, z.B. elliptische Kurven Kryptographie (ECC), erfordert große Rechenkapazität die normalerweise auf von mobilen Geräten bzw. drahtlosen Sensorknoten nicht zur Verfügung steht. Die Implementierung der ECC in Hardware reduziert den Zeit- und Energie-Aufwand. Die Optimierung der Hardware-Implementierungen dient nicht nur der weiteren Reduktion des Zeit- und Energieverbrauches sondern hilft darüber hinaus die Herstellungskosten zu verringern, so dass solche Lösungen auch für kostengünstige Geräte einsetzbar werden.

Im Rahmen dieser Dissertation wurden Optimierungsmöglichkeiten für die Multiplikation der Polynome, die für EC-Operationen eingesetzt werden, untersucht. Ziel der Optimierungen war, dass die Multiplikation mit einer minimalen Anzahl von Additionen (also XOR-Gattern) und Multiplikationen (also AND-Gattern) durchgeführt werden kann. Im Rahmen dieser Arbeit wurde die iterative Bearbeitung von 10 Multiplikations-Methoden (MM) im Gegensatz zur üblichen rekursiven Bearbeitung untersucht. Dabei wurde eine Reihenfolge der Operationen für jede der untersuchten MM ermittelt, die zu einer reduzierten Anzahl von XOR-Operationen führt. Der Einsatz der optimierten Reihenfolge kann die Komplexität der MM wesentlich reduzieren. Zum Beispiel bei der generalisierten Karatsuba-MM [18] beträgt die Reduktion des XOR-Aufwandes durchschnittlich 39 % für Polynom-Längen bis 600 Bits. Für die IHP 0,13µ-Technologie entspricht diese Reduktion des XOR-Aufwandes einer durchschnittlichen Flächen-Reduktion der Polynom-Multiplizierer um 35 %. Bei der 4-Segment-Karatsuba-MM wird nicht nur der XOR-Aufwand, sondern auch die Signal-Verzögerung im Vergleich zur rekursiven Anwendung der originalen Karatsuba-MM reduziert.

Außerdem wurde ein Algorithmus zur Bestimmung einer flächen- und/oder energieoptimalen Kombination der Multiplikations-Methoden entwickelt. Mit dem vorgeschlagenen Algorithmus wurden die flächen- und die energie-optimalen Kombinationen der MM für Polynom-Längen bis 600 Bits bestimmt. Alle ECC-relevanten Polynom-Längen liegen in diesem Bereich. Die durchschnittliche Reduktion der Flächen im Vergleich zu den rekonstruierten Daten aus [30] beträgt 12 %.

Zusätzlich wurde ein energieoptimaler serieller Mehr-Takt-Multiplizierer für 233-Bits Polynome auf Basis Karatsuba-ähnlicher Multiplikations-Methoden entwickelt. Dieser Multiplizierer nutzt die Winograd-MM und basiert auf einen flächenoptimierten 78-Bits-Teil-Multiplizierer. Die theoretischen Ergebnisse wurden mit Hilfe von Synthesedaten für die IHP Technologie erfolgreich verifiziert. Der Energieverbrauch und die Ausführungszeit des Designs sind um 24 % bzw. 28 % kleiner als die des Vergleichsdesigns aus [28].

Kapitel 1

Einführung

Heutzutage gibt es zwei weit verbreitete asymmetrische Kryptosysteme, die den sicheren Datenverkehr über ungesicherte Kanäle gewährleisten können: RSA (Rivest, Shamir, Adleman) und ECC (engl. Elliptic Curve Cryptosystem).

Das RSA-Kryptosystem [1] wurde 1978 veröffentlicht. 1985 wurde das andere asymmetrische Kryptosystem - ECC - von N. Koblitz [2] und von V. Miller [3] unabhängig voneinander entwickelt. Im Vergleich zu den symmetrischen Krypto-Verfahren, wie z. B. dem AES (Advanced Encryption Standard) [4] von NIST (National Institute of Standards and Technology, USA) [5], brauchen asymmetrische Verfahren viel mehr Zeit und Energie für die Ver- und Entschlüsselung der Daten, aber sie gewährleisten mittels der digitalen Signatur die wichtigen Sicherheits-Eigenschaften: Identifikation und Authentisierung der kommunizierenden Seiten, so wie auch die Nichtabstreitbarkeit (engl. non-repudiation). Aus diesem Grund sind die hybriden Systeme für sichere Kommunikation am meisten verbreitet: die RSA- oder ECC-Protokolle sind für die Identifikation (beziehungsweise Authentisierung) der kommunizierenden Seiten und für die Verteilung des symmetrischen Sitzungs-Schlüssels z. B. für AES vorgesehen, mittels dessen dann die Ver- und Entschlüsselung des Daten-Verkehrs geschieht [6]. Eine energie-sparende Anwendung der beiden asymmetrischen Verfahren – RSA und ECC – wurde in [7] veröffentlicht.

Deutlich kürzere ECC-Schlüssel-Längen geben dem ECC-Kryptosystem mehrere Vorteile im Vergleich zu RSA: wesentlich kleinerer Energie- und Zeit-Aufwand bei der Ausführung der kryptographischen Operationen und bei der Übertragung der Ergebnisse und geringere Speicheranforderungen. Diese Eigenschaften machen ECC für die Sicherung der drahtlosen Kommunikation mobiler Geräte und Sensor-Knoten sehr attraktiv.

Trotz dieser Vorteile sind die ECC-Krypto-Operationen immer noch rechen-, zeit- und

energieaufwendig, insbesondere für batterie-/akkubetriebene Geräte mit begrenzten Rechenressourcen. Für solche Geräte ist die Implementierung der kryptographischen Operationen in Hardware oft die einzige Möglichkeit, den gesicherten Datenverkehr zu gewährleisten. Die Optimierung der Implementierung der Krypto-Operationen reduziert die Chip-Fläche und mindert die Herstellungskosten und den Energieverbrauch.

Die ECC basieren auf mathematischen Operationen in endlichen Körpern, oder Galois-Körpern (engl. Galois field, weiter *GF*). Es gibt standardisierte elliptische Kurven (engl. Elliptic Curve, EC) über *GF(p)* und über $GF(2^n)$ [8], [9]. Diese Arbeit wird sich auf binäre EC über $GF(2^n)$ konzentrieren, weil diese für Hardware-Implementierung besser geeignet sind.

Ein wichtiger Aspekt der ECC-Hardware-Implementierung besteht in der Reduzierung der Chip-Fläche des Multiplizierers und den damit verbundenen Herstellungskosten.

Abbildung 1 zeigt die Operationen, auf denen die ECC basiert.

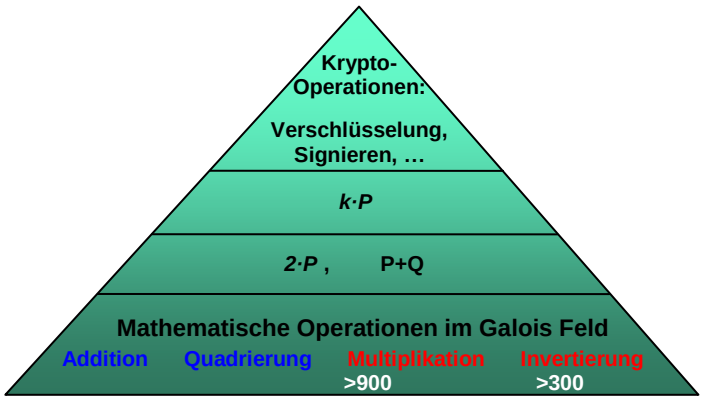


Abbildung 1: ECC-Operationen-Pyramide

Die kryptographischen Protokolle (ECDH, ECAES und ECDSA [10],[11],[12]), die das Generieren des Schlüssels, die Ver- und Entschlüsselung der Daten und die digitalen Signatur-Algorithmen ausführen, beruhen auf der so genannten EC-Punkt-Multiplikation, die als *k·P*-Operation bezeichnet wird. Der Koeffizient *k* ist eine große ganze Zahl und *P(x,y)* ist ein Punkt auf der EC. Die kryptographischen Operationen mit privatem Schlüssel benötigen nur eine Punkt-Multiplikation. Die Signatur-Verifikation und die Daten-Verschlüsselung benötigen zwei *k·P*-

Operationen. Die $k \cdot P$-Operation ist eine komplexe Operation, die als eine Reihe von Punkt-Verdopplungen und Punkt-Additionen durchgeführt werden kann. Alle EC-Punkt-Operationen basieren auf den mathematischen Operationen im Galois Feld.

Die Invertierung und die Multiplikation sind oft durchgeführte und aufwendige Operationen. Zum Beispiel bei einer Verschlüsselung mit einem 233-Bits langen privaten Schüssel, was den Sicherheits-Vorschriften für den Zeitraum 2011 – 2030 entspricht [13], werden durchschnittlich über 300 Inversionen und über 900 Produkte[1] berechnet. Die aufwendigste Operation – die Invertierung – kann durch mehrere Multiplikationen und Quadrierungen ersetzt werden [14]. Diese Arbeit konzentriert sich auf die Optimierung der $GF(2^n)$-Multiplikation.
Die effiziente Hardware-Implementierung der Invertierung, Multiplikation, Quadrierung und Addition der $GF(2^n)$-Elemente kann alle kryptographische Operationen wesentlich beschleunigen.

Momentan existieren mehrere Multiplikations-Methoden (MM) großer ganzer Zahlen, die nach bestimmten Änderungen auch für die Multiplikation von Polynomen geeignet sind. Diese Multiplikations-Methoden stellen die Polynome als Summe von Segmenten dar. Bei der Segmentierung der nm-Bits großen Multiplikanden in n Teile (jeder Teil ist m-Bits groß) wird das Produkt als eine Summe (XOR) der $(2m-1)$-Bits großen Teil-Produkte (TP) berechnet. Die historisch erste – klassische – Multiplikations-Methode [14] benötigt n^2 Teil-Multiplikationen der m-Bits Operanden und $(n-1)^2$ XOR-Operationen der $(2m-1)$-Bits Operanden.
Seit 1962 ist die Karatsuba-Multiplikations-Methode [15] bekannt, die im Vergleich zu der klassischen Multiplikation eine kleinere Anzahl Teil-Multiplikationen, aber eine größere Anzahl Additionen (XOR) der $(2m-1)$-Bits Operanden benötigt. Außerdem sind weitere Additionen (XOR) der m-Bits Operanden notwendig. Bei der Karatsuba-Multiplikations-Methode werden die Multiplikanden immer in 2 Teile ($n=2$) segmentiert. Die Karatsuba-MM wird oft rekursiv für die Multiplikation der 2^k-Segment großen Polynome verwendet.
Seit 1980 ist eine weitere Karatsuba-ähnliche MM, die Winograd Multiplikations-Methode [16] bekannt. Bei dieser Multiplikations-Methode werden die Faktoren in 3 Teile segmentiert. Bei der rekursiven Anwendung dieser Methode können 3^k-Segment große Polynome multipliziert werden. Zwei weitere Multiplikations-Formeln, die eine Segmentierung der Operanden in 3 Terme verwenden, wurden in [17] veröffentlicht.
In 2006 wurde die Karatsuba-Idee für die Multiplikation der n-Segment-großen Operanden generalisiert [18]. In 2004 – 2009 wurden weitere Multiplikations-Methoden für 5-, 6- und 7-Segment große Operanden [19], [20] und für andere, noch größere, Segmentierungen der Operanden in [21], [22] veröffentlicht.

[1] Unter der Anwendung binärer kP-Multiplikation-Methode, affine Koordinaten, polynomiale Basis

Eine andere bekannte MM großer ganzer Zahlen basiert auf der Fourier-Transformation (FT) der Multiplikanden. FT in Galoiskörpern wurde 1971 von J.M. Pollard präsentiert [23]. Obwohl die theoretische Komplexität der FFT im Vergleich zur klassischen oder Karatsuba-MM kleiner ist, hat diese Methode wegen des zusätzlichen Overheads keine praktische Bedeutung für relativ kleine Multiplikanden, also solche mit einer Länge kleiner 1000 Bits [24]. Für *GF(p^n)*, deren Charakteristik die Fermat-Zahl, $p = 2^{2^n} + 1$, ist, ist die FT auch für kleinere Operanden sehr effizient [25]. Da die typische Länge der Multiplikanden bei standardisierten elliptischen Kurven über *GF(2^n)* unter 600 Bits liegt [13], konzentriert sich diese Arbeit nur auf Karatsuba-ähnliche MM.

Ein wichtiger Aspekt der ECC-Hardware-Implementierung besteht in der Reduzierung die Chip-Fläche des Multiplizierers und den damit verbundenen Herstellungskosten. Aus diesem Grund wird die Chip-Fläche des Multiplizierers am häufigsten als Optimierungs-Parameter gewählt. Die wesentliche Reduktion der Chip-Fläche kann mittels der seriellen Ausführung der Multiplikation erreicht werden. Dafür sind alle segmentierungsbasierten MM geeignet.

Eine Kombination der klassischen MM mit den Karatsuba-ähnlichen MM kann die Chip-Parameter des Multiplizierers verbessern [26] – [30]. Die Chip-Fläche und der Energie-Verbrauch des Multiplizieres könnten wesentlich reduziert werden, wenn eine optimale Kombination verschiedener Multiplikations-Verfahren gefunden und verwendet wäre. Eine Analyse der Komplexität der möglichen Kombinationen verschiedener Multiplikations-Methoden mit dem Ziel, algorithmisch die hinsichtlich eines festgelegten Optimierungs-Parameters günstigste Kombination zu bestimmen, fehlt jedoch bis heute.

Im Rahmen dieser Arbeit wurde ein Ansatz zur iterativen Addition der Teil-Produkte entwickelt, der durch die Verringerung der zur Berechnung notwendigen XOR-Gatter zu einer signifikanten Verringerung der Fläche entsprechender Hardware-Implementierungen führt. Die Darstellung der Komplexität der in Hardware implementierten Algorithmen wird im Kapitel 3 diskutiert. Für neun MM wurde ihre Komplexität bei einer iterativen Berechnung der Teil-Produkte bestimmt. Als Optimierungs-Parameter wurde entweder die Chip-Fläche oder der Energie-Verbrauch der Schaltung betrachtet. Als Ergebnis der Analyse wurde die optimale Kombination der MM für alle Polynom-Längen n, mit n≤600 für die 130 nm IHP-Technologie [31] ermittelt. Der vorgeschlagene Algorithmus ist jedoch technologie-unabhängig, weil er herstellerspezifische Flächen-/Energie-Verhältnisse der Gatter berücksichtigen kann. Die Abhängigkeit der optimalen Kombination der MM von der Hersteller-Technologie wurde für alle ECC-relevanten Polynom-Längen untersucht.

Mehrere serielle (Mehr-Takt-) Multiplizierer für 233-Bits-Polynome, die nur einen optimalen kombinatorischen Teil-Multiplizierer beinhalten, wurden miteinander verglichen, um den optimalen Multiplizierer zu bestimmen. Hierfür werden Implementierungs-Kandidaten synthetisiert. Die Synthese-Ergebnisse bestätigen die Qualität der berechneten Flächenmaße. In der Regel sind die Synthese-Ergebnisse sogar noch kleiner.

Diese Arbeit ist folgendermaßen strukturiert:
Kapitel 2 führt die Definition und die bekannten Optimierungen der $GF(2^n)$-Multiplikation ein. Bereits veröffentlichte MM und ihre Kombinationen werden kurz diskutiert.
Kapitel 3 präsentiert die Ideen, die im Rahmen dieser Arbeit für die Implementierung des flächen-/energie-optimalen Polynom-Multiplizierer entwickelt und verwendet wurden:

- Übersichtliche Darstellung der Polynom-Multiplikation
- Die Reduzierung der Anzahl an XOR-Operationen bei einer MM mittels der Berechnung des Produktes nach einer vorbestimmten Reihe der Additionen (XOR), d.h. nach einem vorbestimmten optimierten Ablaufplan
- Eine algorithmische Ermittlung der optimalen Kombination der MM (mit reduzierter Anzahl an XOR-Operationen)

Im Kapitel 4 wird die Bestimmung der Ablaufpläne mit reduzierter Anzahl an XOR-Operationen für Karatsuba-ähnliche MM diskutiert.
Die Kapitel 5 und 6 präsentieren die Ergebnisse dieser Arbeit. Im Kapitel 5 werden die ermittelten Parameter der optimalen kombinatorischen (d.h. 1-Takt) Multiplizierer präsentiert. Die theoretischen Ergebnisse wurden durch die Fläche und den Energie-Verbrauch der synthetisierten Multiplizierer verifiziert. Kapitel 6 zeigt die Parameter der seriellen (d.h. Mehr-Takt-) Multiplizierern für 233-Bits-Polynome, die als die Implementierungs-Kandidaten ausgewählt wurden. Die theoretisch berechneten Parameter der Multiplizierer werden mit den jeweiligen praktischen Werten der synthetisierten Multiplizierer verglichen.
Kapitel 7 bietet eine kurze Zusammenfassung der wichtigsten Ergebnisse sowie eine Diskussion über offene Forschung.

Kapitel 2

Stand der Wissenschaft und Technik

In diesem Kapitel wird eine kurze Einführung in die ECC gegeben, um die Vielzahl der ECC-Optimierungen sowie auch den Schwerpunkt dieser Arbeit zu zeigen. Die ECC basiert auf EC über endlichen Körpern. Für die Hardware-Implementierung sind die EC über erweiterten binären Körpern am besten geeignet. Da die effiziente Hardware-Implementierung der ECC das Ziel dieser Arbeit ist, werden im Folgenden nur EC über diese Körper diskutiert.
Kapitel 2.1 bietet eine kurze Einführung in Mathematik endlicher Körper, um die EC-Punkt-Operationen und deren Optimierungen zu erklären.

2.1. Mathematische Operationen in endlichen Körpern

Ein endlicher Körper ist eine algebraische Struktur. Sie besteht aus einer endlichen Menge ***F*** von Elementen und zwei Operationen: Addition (bezeichnet mit +) und Multiplikation (bezeichnet mit ·).
Für die endlichen Körper gelten die folgenden Eigenschaften:

1 - (***F***, +) ist eine Abelsche Gruppe mit additivem neutralen Element 0, denn
für alle Elemente $a,b \in F$ gelten die folgenden Eigenschaften:

- Abgeschlossenheit: $\forall a,b \in F:\ \ a+b \in F$
- Verknüpfung mit neutralem Element: $\forall a \in F:\ \ a+0=a$
- Existenz eines inversen Elements (Additive Inverse):
  $\forall a \in F\ \ \exists b=-a \in F:\ \ a+b=0$
- Kommutativität: $\forall a,b \in F:\ \ a+b=b+a$
- Assoziativität: $\forall a,b,c \in F:\ \ (a+b)+c=a+(b+c)$

2 - (***F***\{0}, ·) ist eine Abelsche Gruppe mit multiplikativem neutralem Element 1, denn für alle Elemente $a,b \in F$ gelten die folgenden Eigenschaften:

- Abgeschlossenheit: $\forall a,b \in F: \ a \cdot b \in F$
- Verknüpfung mit neutralem Element: $\forall a \in F: \ a \cdot 1 = a$
- Existenz eines inversen Elements (Multiplikative Inverse):
$$\forall a \neq 0 \in F \ \ \exists b = a^{-1} \in F: \ a \cdot b = 1$$
- Kommutativität: $\forall a,b \in F: \ a \cdot b = b \cdot a$
- Assoziativität: $\forall a,b,c \in F: \ (a \cdot b) \cdot c = a \cdot (b \cdot c)$

3 - Distributivgesetz: $\forall a,b,c \in F: \ a \cdot (b+c) = a \cdot b + a \cdot c$.

Endliche Körper werden in der Literatur auch oft Galoiskörper oder Galois Felder genannt und mit *GF* bezeichnet (engl. Galois Field). Auch in dieser Arbeit wird die Bezeichnung *GF* verwendet.

Es gibt EC über den folgenden endlichen Körpern:
- Primkörper *GF(p)*
- erweiterte binäre Körper *GF(2^n)*
- optimal erweiterte Körper *GF(p^n)* (engl. Optimal Extended Fields, OEF) [32].

2.1.1. Primkörper

Primkörper *GF(p)* sind die endlichen Körper mit der Anzahl *p* an Elementen: $(\{0,1,2,\ldots,p-2,p-1\},+,\cdot)$. Die Zahl *p* ist eine Primzahl und heißt Ordnung des Körpers und gewährleistet, dass die Ergebnisse der Operationen mit *GF(p)*-Elementen auch wieder *GF(p)*-Elemente sind. Jeder ganzen Zahl *s*≥*p* entspricht ein Element aus *GF(p)*, das als der Rest der Division der ganzen Zahlen s/p definiert ist. Diese Operation – die Berechnung des Restes der Division – heißt Reduktion, bezeichnet mit $s \bmod p$.

Die **Addition** (**Subtraktion**) der *GF(p)*-Elemente a und b ist als die Addition ganzer Zahlen a und b mit der Reduktion ihrer Summe (Differenz) modulo *p* definiert:

$$\forall a,b \in GF(p): \ a \pm b = (a \pm b) \bmod p$$

Die **Multiplikation** der *GF(p)*-Elemente a und b ist als die Multiplikation ganzer Zahlen a und b mit der Reduktion ihres Produktes modulo *p* definiert:

$$\forall a,b \in GF(p): \ a \cdot b = (a \cdot b) \bmod p$$

Die **Division** der *GF(p)*-Elemente a und b ist als die Multiplikation des *GF(p)*-Elementes a mit dem multiplikativen Inversen zu dem Element b definiert:

$$\forall a,b \in GF(p): \quad \frac{a}{b} = \left(a \cdot b^{-1}\right) \bmod p$$

Ein Beispiel der Primkörper ist der Galoiskörper *GF(2)*. Er besteht aus nur zwei Elementen: {0,1}. Diese Elemente sind als ganze 1-Bit große Binär-Zahlen darstellbar. Alle Operationen im Galoiskörper *GF(2)* sind als Operationen modulo 2 definiert. Deswegen entspricht die Addition so wie auch die Subtraktion der *GF(2)*-Elemente der Booleschen XOR-Funktion der Binär-Zahlen, die im Folgenden durch $\oplus$ dargestellt wird. Die Multiplikation der *GF(2)*-Elemente entspricht der Booleschen AND-Operation.

$$\begin{aligned} \forall a,b \in GF(2): \quad a \pm b &= \left(a \pm b\right) \bmod 2 \quad \rightarrow \quad a \oplus b, \\ \forall a,b \in GF(2): \quad a \cdot b &= \left(a \cdot b\right) \bmod 2 \quad \rightarrow \quad a \wedge b \end{aligned}$$

(1)

2.1.2. Erweiterte binäre Galoiskörper *GF(2^n)*

Es können auch die Galoiskörper *GF(p^n)* mit p^n Elementen konstruiert werden. Die Zahl *p* ist eine Primzahl und wird als Charakteristik des Körpers bezeichnet. Die Zahl *n* ist eine natürliche Zahl. Für die Hardware-Implementierung der ECC sind die erweiterten Galoiskörper der Charakteristik 2 (oder erweiterte binäre Galoiskörper) *GF(2^n)* besonders gut geeignet.

Für jede natürliche Zahl *n* kann ein irreduzibles Polynom ermittelt werden. Irreduzibel bedeutet, dass das Polynom nicht faktorisierbar ist, d.h. es kann nicht als ein Produkt der Polynome vom Grad kleiner *n* dargestellt werden. Mithilfe eines irreduziblen Polynoms $f(t) = 1 \cdot t^n + f_{n-1} \cdot t^{n-1} + \ldots + f_0 \cdot t^0$ mit $f_i \in GF(2)$ kann ein Galoiskörper *GF(2^n)* konstruiert werden.
Die Elemente *A(t)* und *B(t)* eines Galoiskörpers *GF(2^n)* sind Polynome mit einem Grad kleiner *n* mit Koeffizienten aus *GF(2)*:

$$\begin{aligned} A(t) &= a_{n-1} \cdot t^{n-1} + a_{n-2} \cdot t^{n-2} + \ldots + a_0 \cdot t^0, \text{ mit } a_i \in GF(2) \\ B(t) &= b_{n-1} \cdot t^{n-1} + b_{n-2} \cdot t^{n-2} + \ldots + b_0 \cdot t^0, \text{ mit } b_i \in GF(2) \end{aligned}$$

(2)

Die Menge der *GF(2^n)*-Elemente ist dann wie folgt:

$$\left\{0,\ 1,\ t,\ 1+t,\ t^2,\ t^2+1,\ t^2+t,\ t^2+t+1,\ \ldots,\ t^{n-1}+t^{n-2}+\ldots+t+1\right\}.$$

Jedem binären Polynom *S(t)* entspricht ein Element aus *GF(2^n)*, das als Rest der Division der Polynome *S(t)/f(t)* definiert ist. Diese Operation – die Berechnung des Restes der Division der Polynome – heißt Reduktion modulo irreduzibles Polynom, bezeichnet mit $S(t) \bmod f(t)$.

Die **Addition** (**Subtraktion**) der *GF(2^n)*-Elemente ist wie folgt als Addition der Polynome definiert:

$$A(t) \pm B(t) = \left(a_{n-1} \cdot t^{n-1} + \ldots + a_0 \cdot t^0\right) \pm \left(b_{n-1} \cdot t^{n-1} + \ldots + b_0 \cdot t^0\right) = \\ = \left(\left(a_{n-1} \pm b_{n-1}\right) \bmod 2\right) \cdot t^{n-1} + \ldots + \left(\left(a_0 \pm b_0\right) \bmod 2\right) \cdot t^0 \quad \textbf{(3)}$$

Die **Multiplikation** zweier *GF(2^n)*-Elemente $A(t)$ und $B(t)$ ist als Multiplikation zweier Polynome mit anschließender Reduktion des Polynom-Produktes *C(t)* mit dem irreduziblen Polynom *f(t)* definiert:

$$C'(t) = \left(A(t) \cdot B(t)\right) \bmod f(t) = C(t) \bmod f(t) \quad \textbf{(4)}$$

D.h. dass die Multiplikation **(4)** aus den folgenden zwei Schritten besteht:

Schritt 1 – *Berechnung des Polynoms C(t)* vom Grad *(2n-2)*:

$$C(t) = A(t) \cdot B(t) = \left(a_{n-1} \cdot t^{n-1} + \ldots + a_0 \cdot t^0\right) \cdot \left(b_{n-1} \cdot t^{n-1} + \ldots + b_0 \cdot t^0\right) = \\ = c_{2n-2} \cdot t^{2n-2} + \ldots + c_0 \cdot t^0$$

mit $$c_i = \left(\sum_{\substack{n > j \geq 0, \\ n > k \geq 0, \\ j+k=i}} a_j \cdot b_k \right) \bmod 2 \quad \textbf{(5)}$$

Schritt 2 – *Reduktion* des *C(t)*-Polynoms:

$$C'(t) = C(t) \bmod f(t) \quad \textbf{(6)}$$

Die **Division** zweier Polynome $A(t)$ und $B(t)$ aus *GF(2^n)* ist als Multiplikation des Dividenden mit dem multiplikativen Inversen des Divisors definiert:

$$\frac{A(t)}{B(t)} = \left(A(t) \cdot \left(B(t)^{-1} \bmod f(t)\right)\right) \bmod f(t) \quad \textbf{(7)}$$

Das multiplikative Inverse zu einem Element aus *GF(2^n)* wird normalerweise mittels des erweiterten Euklidischen Algorithmus [14], [33] ermittelt. Unter Verwendung des kleinen Fermat Satzes [14] kann die Invertierung durch mehrere Multiplikationen folgendermaßen ersetzt werden:

$$B(t)^{-1} = B(t)^{2^n-2} \bmod f(t) \qquad \textbf{(8)}$$

2.2. Elliptische Kurven über *GF(2^n)*

Die für die kryptographische Anwendung geeigneten standardisierten EC über *GF(2^n)* [8] sind EC entsprechend Gleichung **(9)**:

$$y^2 + xy = x^3 + ax^2 + b \text{ mit } a,b \in GF(2^n) \qquad \textbf{(9)}$$

Jedes Paar der Elemente aus *GF(2^n),* das die Gleichung **(9)** erfüllt, ist ein Punkt *P=(x,y)* der elliptischen Kurve ***E*** (weiter EC-Punkt). Eine endliche Menge solcher Paare erfüllt Gleichung **(9)**. Zusätzlich zu diesen Paaren wird der Punkt ***O*** im Unendlichen definiert. Die Menge der Punkte der elliptischen Kurve ***E*** besteht aus der Menge aller Paare *(x, y)*, die die Gleichung der EC **(9)** erfüllen, und des unendlichen Punktes ***O***. Die Menge der Punkte der EC ***E*** bildet eine Abelsche Gruppe (***E***, +) bzgl. Addition mit additivem neutralem Element ***O***. Für alle Elemente $P,Q \in E$ gelten folgende Eigenschaften:

- Abgeschlossenheit: $\forall P,Q \in E: \quad P+Q \in E$
- Verknüpfung mit neutralem Element: $\forall P \in E: \quad P+O=P$
- Existenz eines inversen Elements (Additive Inverse):
  $$\forall P \in E \quad \exists Q=-P \in E: \quad P+Q=O$$
  Das additive Inverse für *P=(x$_1$,y$_1$)*$\in$***E*** ist folgendermaßen definiert:
  $$-P=(x_1, x_1+y_1)$$
- Kommutativität: $\forall P,Q \in E: \quad P+Q=Q+P$
- Assoziativität: $\forall P,Q,S \in E: \quad (P+Q)+S=P+(Q+S)$

Die **Addition** zweier EC-Punkte ist folgendermaßen definiert [33]:

- Die Addition zweier verschiedener EC-Punkte (**Punkt-Addition**):
  P=(x$_1$,y$_1$), Q=(x$_2$,y$_2$) $\in$***E*** mit *P*$\neq\pm$*Q* (d.h. mit *x$_1$*$\neq$*x$_2$*):

$$R = P + Q = (x_3, y_3) \quad \text{mit} \quad x_3 = \left(\frac{y_2 + y_1}{x_2 + x_1}\right)^2 + \frac{y_2 + y_1}{x_2 + x_1} + x_1 + x_2 + a$$

$$y_3 = \frac{y_2 + y_1}{x_2 + x_1}(x_1 + x_3) + x_3 + y_1$$

(10)

- Die Addition zweier gleicher EC-Punkte (**Punkt-Verdopplung**):
  $P=(x_1,y_1) \in \boldsymbol{E}$ mit $x_1 \neq 0$:

$$R = P + P = 2 \cdot P = (x_3, y_3) \quad \text{mit} \quad x_3 = x_1^{\,2} + \frac{b}{x_1^{\,2}}$$

$$y_3 = x_1^{\,2} + \left(x_1 + \frac{y_1}{x_1}\right) \cdot x_3 + x_3$$

(11)

Die *k*-fache Addition eines EC-Punktes $P \in \boldsymbol{E}$, wobei *k* eine natürliche Zahl ist, wird die EC-Punkt-Multiplikation (oder Skalarmultiplikation des EC-Punktes) genannt, mit ***kP*** bezeichnet:

$$k \cdot P = \underbrace{P + P + \ldots + P}_{k \text{ Summanden}}$$

(12)

Die minimale Zahl $r: \; r \cdot P = O$ ist Ordnung des EC-Punktes $P \in \boldsymbol{E}$.

Die Gleichung der EC **(9)**, das irreduzible Polynom, ein gewählter Punkt *G* (Basis-Punkt, engl. *base point G*) der EC ***E*** und dessen Ordnung sind die Parameter der EC.

2.3. Übersicht der ECC-Optimierungen

Die EC-Punkt-Multiplikation **(12)** wird in allen ECC-Protokollen genutzt. Die ECC-Operationen mit dem privaten Schlüssel (z.B. das Generieren der digitalen Signatur und die Entschlüsselung) benötigen eine EC-Punkt-Multiplikation, die den größten Anteil am Aufwand dieser Operation hat. Die ECC-Operationen mit dem öffentlichen Schlüssel (z.B. die Verschlüsselung der Daten oder die Verifikation der Signatur) bestehen aus zwei ***kP***-Operationen. Der private Schlüssel ist eine *n*-Bits große Zufallszahl *k*, die nur dem Besitzer des Schlüsselpaares bekannt ist. Eine Länge von 233 Bits gilt als sichere Länge[1] *n* des Schlüssels *k* für die Zeitperiode 2011-2030. Der öffentliche Schlüssel besteht aus den Parametern der EC und einem EC-Punkt *S*, der durch Skalarmultiplikation des Basis-Punkt *G* mit dem privaten Schlüssel *k* folgendermaßen berechnet wird:

[1] Entsprechend den Sicherheits-Hinweisen aus [8] für die elliptische Kurve B-233

$$S = k \cdot G \tag{13}$$

Der öffentliche Schlüssel wird bekannt gegeben.

Obwohl der Zusammenhang der beiden Punkte *S* und *G* und ihre Koordinate bekannt sind, ist die Aufgabe, den privaten Schlüssel *k* einer Person zu berechnen, komplex. Sie wird als Problem des diskreten Logarithmus auf elliptischen Kurven (*elliptic curve discrete logarithm problem,* ECDLP) bezeichnet.

Eine effiziente Ausführung der ***kP***-Operation bestimmt die Effizienz der ECC. In diesem Kapitel werden nicht alle Optimierungs-Möglichkeiten ausführlich erklärt, sondern die Vielzahl der Optimierungs-Möglichkeiten auf jeder Schicht der ECC-Operations-Pyramide aufgezeigt. **Abbildung 2** gibt eine Übersicht der Optimierungen.

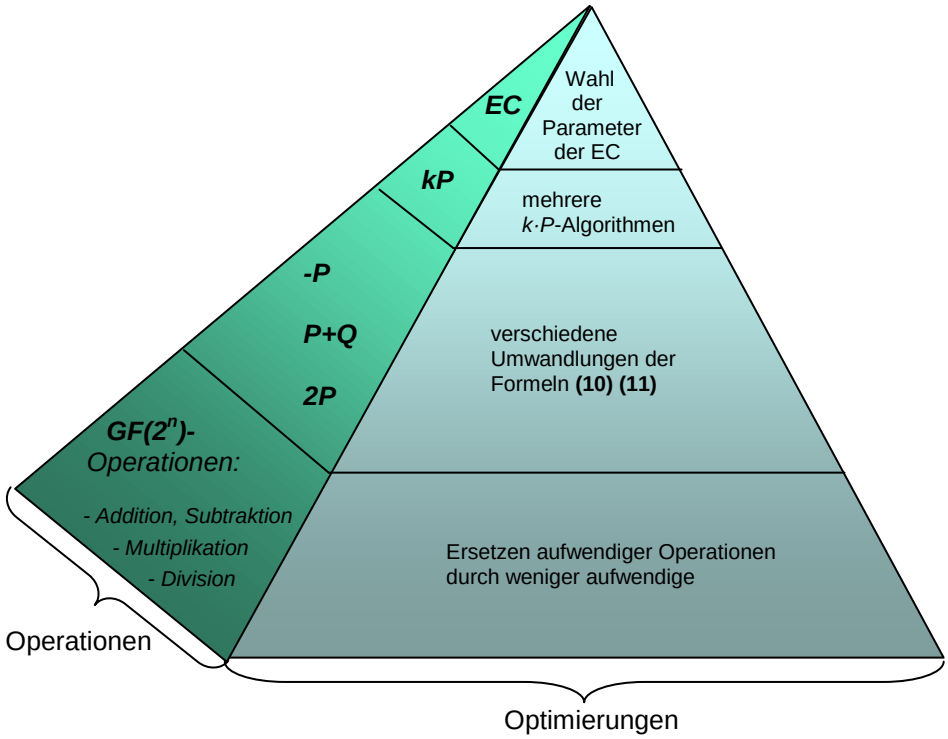


Abbildung 2: Optimierungs-Möglichkeiten der EC-Operationen

Einige Optimierungen wurden bei der Standardisierung der EC bereits durchgeführt: die Wahl der Koeffizienten in der Gleichung **(9)** und des irreduziblen Polynomes *f(t)*. Ein Beispiel sind die Koblitz-Kurven: K-163, K-233, K-283, K-409

und K-571 [8]. Ihre Koeffizienten in Gleichung **(9)** sind *a=0* (oder *a=1*) und *b=1*. Auch bei den elliptischen Kurven B-163, B-233, B-283, B-409 und B-571 [8] ist *a=1*. So wird die Komplexität der EC-Punkt-Addition (siehe **(10)**) und der EC-Punkt-Verdopplung (siehe **(11)**) reduziert. Aus Effizienz-Gründen wurde jeweils ein Trinom oder ein Pentanomial als irreduzibles Polynom gewählt.

Weitere Optimierungen beziehen sich auf den Ausführungs-Algorithmus der *kP*-Operation **(12)**. Der Basis-Algorithmus ist als binäre oder „double and add"-Methode der *kP*-Berechnung bekannt [33] und stellt die Zahl *k* als eine Binärzahl dar: $k=k_{n-1}\cdot 2^{n-1}+ k_{n-2}\cdot 2^{n-2}+\ldots+ k_0\cdot 2^0$, mit $k_i=\{0,1\}$. In diesem Fall benötigt die *kP*-Operation (*n-1*) Punkt-Verdopplungen nach Gleichung **(11)** und *H(k)* Punkt-Additionen nach Gleichung **(10)**, wobei *H(k)* das Hammings-Gewicht der Binärzahl *k* ist. Die verschiedenen Techniken basieren auf der Zerlegung der Zahl *k* in mehrere kleinere, *w*-Bits große Teile (*windows*). Die Produkte aller möglichen *w*-Bits großen Zahlen mit dem EC-Punkt *P* müssen vorberechnet und für die weitere Nutzung gespeichert werden. Formel **(14)** zeigt dies:

$$k \cdot P = \underbrace{k_{n-1}k_{n-2}k_{n-3}}_{w_{n/3-1}} \ldots \underbrace{k_5k_4k_3}_{w_1}\underbrace{k_2k_1k_0}_{w_0} \cdot P = \left(w_{n/3-1} \cdot 2^{(n/3-1)\cdot 3} \ldots + w_1 \cdot 2^{1\cdot 3} + w_0\right)\cdot P =$$
$$= 2^{(n/3-1)\cdot 3} \cdot \left(w_{n/3-1} \cdot P\right) + \ldots + 2^{1\cdot 3} \cdot \left(w_1 \cdot P\right) + w_0 \cdot P \qquad \textbf{(14)}$$

Die Zahl *k* ist in **(14)** als die Summe der 3-Bits großen *windows* (*w=3*) dargestellt. Die optimierte Berechnung $2^i\cdot P$ wurde in [34] veröffentlicht. Weitere Darstellungs-Techniken der Zahl *k*, wie z.B. NAF (*non-adjacent form*), sind in [33] beschrieben.

Eine andere Technik zur *kP*-Berechnung ist die Montgomery-Multiplikation [35]. Bei dieser Methode wird die *x*-Koordinate von *kP* nur aus den *x*-Koordinaten aller Punk-Verdopplungen und Punkt-Additionen ermittelt. Die Berechnung der *y*-Koordinate bei allen Punkt-Verdopplungen und Punkt-Additionen kann somit eingespart werden. Die Methode der Wiederherstellung der *y*-Koordinate von *kP* wurde in [36] veröffentlicht.

Eine wesentliche Reduktion der Komplexität der kP-Operation bringt der Übergang zu projektiven Koordinaten in **(10)** und **(11)**, weil viele Divisionen der $GF(2^n)$-Elemente vermieden werden können. Die Montgomery-kP-Multiplikation in projektiven Lopez-Dahab-Koordinaten [36] benötigt zum Beispiel nur eine Division der $GF(2^n)$-Elemente.

Einige dieser Optimierungen sind für Software-Implementierung besser geeignet, wie z.B. die kP-Berechnung mit Speichern mehrerer vorberechneter Werte. Andere Optimierungen, wie z.B. die Montgomery-EC-Punkt-Multiplikation unter

Verwendung der Lopez-Dahab-Koordinaten, können Software- als auch Hardware-Implementierungen von ECC wesentlich beschleunigen.

Auch verschiedene Umwandlungen der Formel für die EC-Punkt-Addition **(10)** und der Formel für die EC-Punkt-Verdopplung **(11)** können die Software- und die Hardware-Implementierung wesentlich verbessern. Diese Formeln können arithmetisch so umgewandelt werden, dass die Anzahl der aufwendigsten Operationen – der Divisionen und/oder der Multiplikationen der *GF(2^n)*-Elemente – reduziert wird. Trotz der steigenden Anzahl anderer Operationen kann die neue Reihe der Berechnungen vorteilhaft sein. Ein Beispiel solcher Umwandlungen wird in [37] vorgeschlagen. Die Ablaufpläne **(15)** und **(16)** zeigen den Unterschied. Die Berechnung der Koordinaten aus Gleichung **(11)** nach dem Ablaufplan **(15)** benötigt 2 Divisionen, 1 Multiplikation, 1 Quadrierung und 4 Additionen der *GF(2^n)*-Elemente:

$\alpha_1 = x_1^2$	-	1 Quadrierung
$\lambda = x_1 + \frac{y_1}{x_1}$	-	1 Division und 1 Addition
$x_3 = \alpha_1 + \frac{b}{\alpha_1}$	-	1 Division und 1 Addition
$y_3 = \alpha_1 + \lambda \cdot x_3 + x_3$	-	1 Multiplikation und 2 Additionen

(15)

Die Berechnung der Koordinaten aus Gleichung **(11)** nach dem Ablaufplan **(16)** (entsprechend [37]) benötigt 1 Division, 1 Multiplikation, 2 Quadrierungen und 5 Additionen:

$\lambda = x_1 + \frac{y_1}{x_1}$	-	1 Division und 1 Addition
$x_3 = \lambda^2 + \lambda + a$	-	1 Quadrierung und 2 Addition
$y_3 = x_1^2 + (\lambda + 1) \cdot x_3$	-	1 Quadrierung, 1 Multiplikation und 2 Additionen

(16)

Der Ablaufplan **(16)** benötigt 1 Division weniger, aber 1 Quadrierung und 1 Addition mehr als Ablaufplan **(15)**. Da die Division eine viel komplexere Operation als die beiden anderen ist, ist der in [37] vorgeschlagene Ablaufplan vorteilhaft.

Die Division ist die aufwendigste Operation in *GF(2^n)*. Die Division *a/b* wird meistens als Produkt von a mit dem Inversen von *b* berechnet. Mehrere Algorithmen implementieren die Suche des Inversen: erweiterten euklidischen Algorithmus [14], [33] (engl. extended Euclidean algorithm, EEA), Almost Inverse Algorithm (AIA) [37], oder Modified AIA [38] oder auch der kleine Fermat Satz [14] für die *GF(2^n)*:

$$x^{2^n} \mod f(t) \equiv x \qquad \Rightarrow \quad x^{2^n-2} \mod f(t) \equiv x^{-1} \tag{17}$$

In [39] wurde die Division zweier *GF(2^n)*-Elemente auf Basis von MAIA vorgeschlagen.

Die Optimierungen der *GF(2^n)*-Operationen (siehe **Abbildung 2**) basieren auf dem Ersetzen aufwendiger Operationen durch weniger aufwendige Operationen. Z.B. kann die Invertierung durch mehrere Multiplikationen und Quadrierungen nach Formel **(17)** ersetzt werden. Damit kann bei der Hardware-Implementierung des EC-Prozessors über 15% der Chip-Fläche gespart werden [40]. Obwohl die Quadrierung ein Sonderfall der Multiplikation ist, kann sie mittels eines eigenen, viel schnelleren und einfacheren Algorithmus umgesetzt werden [33], und wird aus diesem Grund oft als eine selbstständige Operation betrachtet.
Damit kann die *kP*-Operation mittels nur dreier mathematischer Operationen in *GF(2^n)* implementiert werden: Multiplikation, Quadrierung und Addition. Die Multiplikation der *GF(2^n)*-Elemente ist im Vergleich mit Addition (Subtraktion) und Quadrierung die aufwendigste Operation. Die optimale Implementierung der Multiplikation kann die *kP*-Operation beschleunigen und auch den Energie-Verbrauch, die Chip-Fläche und die damit verbundenen Herstellungs-Kosten der Schaltung wesentlich reduzieren. Dieser Effekt resultiert aus der Tatsache, dass die Multiplikation nicht nur die aufwendigste, sondern auch die am häufigsten ausgeführte Operation ist. Diese Arbeit konzentriert auf die Multiplikation.

Im nächsten Kapitel werden *GF(2^n)*-Elemente als ganze Binär-Zahlen dargestellt. Die klassische MM ganzer Binär-Zahlen und deren Optimierungen werden auch diskutiert, weil sie für die Polynom-Multiplikation der *GF(2^n)*-Elemente angepasst werden können.

2.4. Binäre Darstellung der Polynome

Diese Arbeit konzentriert sich auf Schritt 1 der Multiplikation zweier *GF(2^n)*-Elemente, d.h. nur auf die Optimierung der Berechnung des Polynom-Produktes **(5)**. Da die *GF(2^n)*-Elemente als Bits-Strings dargestellt werden können, können für die Berechnung des Polynom-Produktes die Optimierungs-Methoden der Multiplikation großer Binär-Zahlen verwendet werden. In diesem Kapitel werden die *GF(2^n)*-Elemente binär dargestellt. Kapitel 2.5 gibt eine Übersicht der Multiplikations-Methoden großer ganzer binärer Zahlen.

Die Koeffizienten der *GF(2^n)*-Elemente *A(t)* und *B(t)* sind Elemente aus *GF(2)* und können entweder 0 oder 1 werden. Daraus folgt, dass die Elemente $A(t)$ und $B(t)$ als binäre, *n*-Bits große Zahlen *A* und *B* dargestellt werden können:

$$A(t) = a_{n-1} \cdot t^{n-1} + \ldots + a_1 \cdot t^1 + a_0 \cdot t^0 \quad \rightarrow \quad A = a_{n-1}a_{n-2}...a_1a_0$$
$$B(t) = b_{n-1} \cdot t^{n-1} + \ldots + b_1 \cdot t^1 + b_0 \cdot t^0 \quad \rightarrow \quad B = b_{n-1}b_{n-2}...b_1b_0$$

(18)

Die binär dargestellten *GF(2^n)*-Elemente werden weiterhin *n*-Bits große Polynome genannt. Die Addition und die Subtraktion der *GF(2^n)*-Elemente **(3)** entspricht der bitweisen XOR-Operation der *n*-Bits großen Polynome und wird weiter mit „⊕“ bezeichnet: *A(t)±B(t)* → *A⊕B.* Um Missverständnisse zu vermeiden, wird weiterhin das Symbol $\bigoplus_{i=0}^{n}$ anstatt des Symbols $\sum_{i=0}^{n}$ bei den binär dargestellten *GF(2^n)*-Elementen benutzt:

$$A(t) \quad \rightarrow \quad A = a_{n-1}a_{n-2}...a_1a_0 = \bigoplus_{i=0}^{n-1} a_i \cdot 2^i$$
$$B(t) \quad \rightarrow \quad B = b_{n-1}b_{n-2}...b_1b_0 = \bigoplus_{i=0}^{n-1} b_i \cdot 2^i$$
$$C(t) \quad \rightarrow \quad C = c_{2n-2}c_{2n-3}...c_1c_0 = \bigoplus_{i=0}^{2n-2} c_i \cdot 2^i$$

(19)

Das Symbol „⊕“ wird im Rahmen dieser Arbeit auch für die Bezeichnung der XOR-Operation mehrerer Operanden (ähnlich dem Summenzeichen) verwendet:

$$a_0 \oplus a_1 \oplus a_2 \oplus ... \oplus a_n = \bigoplus_{i=0}^{n} a_i$$

(20)

Mit den Bezeichnungen **(19)** und **(20)** kann die Berechnung des Polynom-Produktes **(5)** folgendermaßen dargestellt werden:

$$C(t) = A(t) \cdot B(t) \quad \rightarrow \quad C = A \cdot B = \left(\bigoplus_{i=0}^{n-1} a_i \cdot 2^i \right) \cdot \left(\bigoplus_{i=0}^{n-1} b_i \cdot 2^i \right) = \bigoplus_{i=0}^{2n-2} c_i \cdot 2^i ,$$
$$\text{mit } c_i = \bigoplus_{\substack{n>j\geq 0,\\ n>k\geq 0,\\ j+k=i}} a_j \cdot b_k$$

(21)

Wenn *A* und *B* nicht die binär dargestellten *n*-Bits-Polynome sind, sondern die binär dargestellten ganzen (n-Bits großen) Zahlen, ist ihr Produkt folgendermaßen definiert:

$$C = A \cdot B = \left(\sum_{i=0}^{n-1} a_i \cdot 2^i \right) \cdot \left(\sum_{i=0}^{n-1} b_i \cdot 2^i \right) = \sum_{i=0}^{2n-2} c_i \cdot 2^i \text{ , mit } \qquad c_i = \sum_{\substack{n>j\geq 0,\\ n>k\geq 0,\\ j+k=i}} a_j \cdot b_k$$

(22)

Aus dem Vergleich der Formeln **(21)** und **(22)** folgt: die Multiplikations-Methoden ganzer *n*-Bits großer Binär-Zahlen können für die Multiplikation der *n*-Bits-

Polynome angepasst werden. Dafür müssen alle Additionen in **(22)** durch die XOR-Operation ersetzt werden.

2.5. Multiplikation großer ganzer Binär-Zahlen

Die Multiplikation großer ganzer Binär-Zahlen ist kein neues Problem. Bei Software-Implementierungen der Multiplikation werden die Faktoren in 1-Wort große Segmente geteilt, weil alle CPUs die 1-Wort- (*m*-Bits-) Operanden in einem Takt verarbeiten können. Das Produkt wird als eine Summe der Segment-Produkte (Teil-Produkte) berechnet.

Bei Hardware-Implementierungen kann diese Idee auch verwendet werden. Die großen Faktoren werden als Summe der Segmente dargestellt. Die beiden Operanden können in gleich viele aber auch in unterschiedlich viele Segmente geteilt werden. In diesem Kapitel werden die MM, die auf einer Segmentierung der Operanden basieren, beschrieben. Diese werden im Folgenden als segmentierungs-basierte MM bezeichnet.

2.5.1. Klassische Multiplikations-Methode und ihre Optimierungen

Die klassische Multiplikations-Methode kann für gleich oder für unterschiedlich segmentierte Multiplikanden verwendet werden. Formel **(23)** zeigt die klassische Multiplikation zweier Binär-Zahlen gleicher Länge *n* bei Teilung eines Multiplikanden in n_1 (m_1-Bits große) Segmente und des anderen Multiplikanden in n_2 (m_2-Bits große) Segmente, $n=n_1m_1=n_2m_2$.

$$A \cdot B = \left(\underbrace{a_{n_1 \cdot m_1 - 1} \ldots a_{(n_1-1)m_1}}_{A_{n_1-1}} \ldots \underbrace{a_{m_1-1} \ldots a_0}_{A_0} \right) \cdot \left(\underbrace{b_{n_2 \cdot m_2 - 1} \ldots b_{(n_2-1)m_2}}_{B_{n_2-1}} \ldots \underbrace{b_{m_2-1} \ldots b_0}_{B_0} \right) =$$
$$= \left(A_{n_1-1} \cdot 2^{(n_1-1)m_1} + \ldots + A_1 \cdot 2^{m_1} + A_0 \right) \cdot \left(B_{n_2-1} \cdot 2^{(n_2-1)m_2} + \ldots + B_1 \cdot 2^{m_2} + B_0 \right) \quad \textbf{(23)}$$

Jeder Term aus der 1. Klammer wird mit jedem Term aus der 2. Klammer multipliziert. Das Produkt wird als der Summe der Produkte der Terme (Teil-Produkte) gesucht.

Verschiedene Optimierungs-Ansätze der klassischen MM sind in **Abbildung 3** gezeigt.

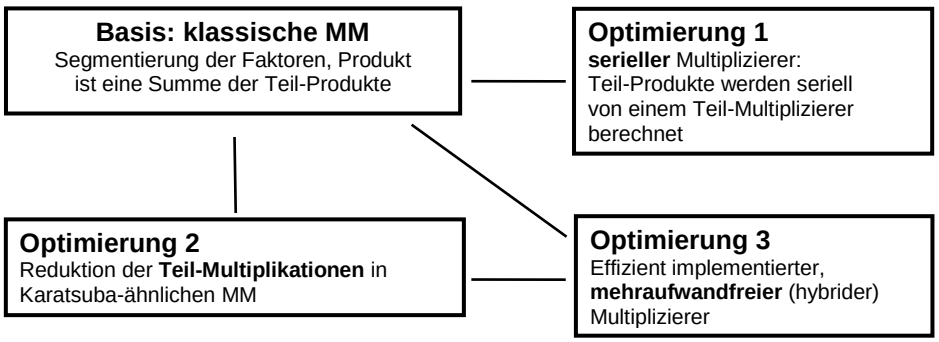


Abbildung 3: Optimierungs-Ansätze der klassischen MM

Eine Art der Optimierung des *n*-Bits Multiplizierers ist die Reduktion der Chip-Fläche mittels serieller Mehr-Takt-Ausführung der Multiplikation (siehe **Abbildung 3**, Optimierung 1). Je Takt wird ein Teil-Produkt ermittelt. Die Komplexität der MM wird dabei nicht reduziert.
Die Karatsuba-ähnlichen MM reduzieren die Anzahl der unterschiedlichen Teil-Multiplikationen (siehe **Abbildung 3**, Optimierung 2). Die Anzahl der Additionen steigt dabei an. Eine weitere Optimierungs-Möglichkeit besteht in der Verringerung der Additionen und der Suche nach einer günstigen Kombination der Karatsuba-ähnlichen MM und der klassischen MM (siehe **Abbildung 3**, Optimierung 3).

Als gründlich untersucht kann nur die Optimierung 2 – die Reduktion der Anzahl der Teil-Produkte – bezeichnet werden, obwohl das immer noch nicht abgeschlossen ist. Viele Veröffentlichungen zum Thema „overlap-free“ Karatsuba-Multiplikation und über die Suche des optimalen hybriden Multiplizierers sind in der letzten Zeit erschien. Die folgenden Abschnitte stellen jeden Optimierungs-Ansatz einzeln dar.

2.5.1.1. Serielle Ausführung der klassischen Multiplikation

Die Anwendung der Segmentierung der Faktoren gibt die Möglichkeit, die Multiplikation seriell, also in mehreren Takten, unter Verwendung eines kleineren Teil-Multiplizierers, auszuführen.

Ein typisches Beispiel dafür ist die Teilung nur eines *n*-Bits Multiplikanden in *n* Segmente [33]. Der andere Multiplikand bleibt unzerlegt. Der Fall entspricht der Formel **(23)** mit $n_1=n$, $m_1=1$, $n_2=1$, $m_2=n$. Formel **(24)** stellt diesen Fall dar.

$$A \cdot B = \left(a_{n-1} \cdot 2^{n-1} + \ldots + a_1 \cdot 2^1 + a_0\right) \cdot \left(b_{n-1} b_{n-2} \ldots b_1 b_0\right) =$$
$$= \underbrace{\underbrace{a_{n-1} \cdot b_{n-1} b_{n-2} \ldots b_1 b_0}_{\text{Teil-Multiplikation}} \cdot 2^{n-1} + \ldots + \underbrace{a_1 \cdot b_{n-1} b_{n-2} \ldots b_1 b_0}_{\text{Teil-Multiplikation}} \cdot 2^1 + \underbrace{a_0 \cdot b_{n-1} b_{n-2} \ldots b_1 b_0}_{\text{Teil-Multiplikation}}}_{n \text{ Teil-Produkte}}$$

(24)

Die Fläche eines Teil-Multiplizierers, der eine Teil-Multiplikation in **(24)** ausführt, besteht aus nur $m_2=n$ AND-Gattern. Die Fläche eines kombinatirischen Multiplizierers (d.h. der 1-Takt-Schaltung) besteht aus den Flächen aller n Teil-Multiplizierer, d.h. aus n^2 AND-Gattern und $(n-1)^2$ Addierern.

Wenn Formel **(24)** als eine n-Takt-Schaltung implementiert wird, besteht die Fläche des ganzen Multiplizierers nur aus der Fläche eines Teil-Multiplizierers ($m_2=n$ AND-Gattern) und einer weiteren Einheit, die die ermittelten Teil-Produkte akkumuliert. Bei jedem Takt wird nur 1 Teil-Produkt $a_i \cdot B$ berechnet. Das ganze Produkt kann innerhalb der n Takte in den Registern akkumuliert werden. Die Schaltung der Multiplikation besteht aus nur einem Teil-Multiplizierer, einer Akkumulationseinheit, zu der die Addierer und die Akkumulations-Register gehören, und einer Steuerungs-Einheit. Die Komplexität der Multiplikation ist damit nicht verringert, aber die mehrfache Anwendung desselben Teil-Multiplizierers reduziert wesentlich die Chip-Fläche des Multiplizierers. Ein weiterer Vorteil dieser Segmentierung ist die Möglichkeit, die Reduktion des Teil-Produktes einzubauen [41].

In [42] sind verschiedene Techniken der Teilung der beiden Multiplikanden beschrieben, die für die Software-Implementierung großer ganzer Zahlen verwendet werden.

2.5.1.2. Verringerung der Teil-Multiplikationen

Die erste MM mit reduzierter Anzahl der unterschiedlichen Teil-Produkte ist die Karatsuba-MM [15]. Diese MM und mehrere weitere Karatsuba-ähnliche Multiplikations-Methoden gehen davon aus, dass die Segmentierung der beiden Operanden gleich ist. Jeder binäre Multiplikand wird in 2-, 3-, oder mehr Segmente geteilt. Die klassische Multiplikations-Formel bei der Zerlegung der beiden Faktoren in n Segmente (siehe Formel **(23)** mit $n_1= n_2=n$, $m_1=m_2=m$) ist wie folgt:

$$A \cdot B = \left(\sum_{i=0}^{n-1} A_i \cdot 2^{i \cdot m}\right) \cdot \left(\sum_{i=0}^{n-1} B_i \cdot 2^{i \cdot m}\right) = \sum_{i=0}^{2n-2} CS_i \cdot 2^{i \cdot m} \text{, mit} \quad CS_i = \sum_{\substack{n > j \geq 0, \\ n > k \geq 0, \\ j+k=i}} A_j \cdot B_k$$

(25)

Formel **(26)** stellt die Zerlegung der Multiplikanden in 2 Segmente und die Berechnung des Produktes nach der klassischen Multiplikations-Methode **(25)** dar:

$$A \cdot B = \left(A_1 \cdot 2^m + A_0\right) \cdot \left(B_1 \cdot 2^m + B_0\right) = A_1 B_1 \cdot 2^{2m} + \left(A_0 B_1 + A_1 B_0\right) \cdot 2^m + A_0 B_0$$

(26)

Hier sind 4 verschiedene Teil-Produkte (TP) zu berechnen: A_0B_0, A_0B_1, A_1B_0 und A_1B_1. Wenn die Segmente A_i und B_i groß sind, kann die Zerlegung wieder durchgeführt werden. So kann die Formel **(26)** mehrmals bis zu 1-Bit großen Operanden rekursiv verwendet werden.

Mittels Karatsuba Idee kann die Berechnung eines TP gespart werden, wenn das Produkt folgendermaßen berechnet wird:

$$\begin{aligned} A \cdot B &= \left(A_1 \cdot 2^m + A_0\right) \cdot \left(B_1 \cdot 2^m + B_0\right) = \\ &= A_1 B_1 \cdot 2^{2m} + \left(\left(A_0 + A_1\right)\left(B_0 + B_1\right) - A_0 B_0 - A_1 B_1\right) \cdot 2^m + A_0 B_0 \end{aligned}$$

(27)

Das Paar der Teil-Produkte (A_0B_1+A_1B_0) aus **(26)** wurde von Karatsuba folgendermaßen dargestellt:

$$\left(A_0 B_1 + A_1 B_0\right) = \left(A_0 + A_1\right)\left(B_0 + B_1\right) - A_0 B_0 - A_1 B_1$$

(28)

Die Karatsuba-Multiplikations-Formel **(27)** beinhaltet nur 3 verschiedene Teil-Produkte: A_0B_0, A_1B_1 und $(A_0+A_1)(B_0+B_1)$, aber mehr Additionen. Sie kann auch mehrmals rekursiv angewendet werden.

Die gleiche Idee funktioniert auch bei der Zerlegung der Multiplikanden in 3 Segmente. Die klassische Multiplikations-Formel (siehe **(25),** mit $n=3$) ist wie folgt:

$$\begin{aligned} A \cdot B = \left(A_2 \cdot 2^{2m} + A_1 \cdot 2^m + A_0\right) \cdot \left(B_2 \cdot 2^{2m} + B_1 \cdot 2^m + B_0\right) = \underbrace{A_2 B_2}_{=CS_4} \cdot 2^{4m} + \\ + \underbrace{\left(A_2 B_1 + A_1 B_2\right)}_{=CS_3} \cdot 2^{3m} + \underbrace{\left(A_2 B_0 + A_1 B_1 + A_0 B_2\right)}_{=CS_2} \cdot 2^{2m} + \underbrace{\left(A_1 B_0 + A_0 B_1\right)}_{=CS_1} \cdot 2^m + \underbrace{A_0 B_0}_{=CS_0} \cdot 2^0 \end{aligned}$$

(29)

In [16] und [43] wurde die folgende alternative Formel veröffentlicht:

$$\begin{aligned} A \cdot B = (A_2 \cdot 2^{2m} + A_1 \cdot 2^m + A_0 \cdot 2^0) \cdot (B_2 \cdot 2^{2m} + B_1 \cdot 2^m + B_0 \cdot 2^0) = \\ = \underbrace{A_2 B_2}_{=CS_4} \cdot 2^{4m} + \underbrace{\left(\left(A_2 + A_1\right)\left(B_2 + B_1\right) - A_1 B_1 - A_2 B_2\right)}_{=CS_3} \cdot 2^{3m} + \\ + \underbrace{\left(\left(A_2 + A_0\right)\left(B_2 + B_0\right) - A_0 B_0 - A_2 B_2 + A_1 B_1\right)}_{=CS_2} \cdot 2^{2m} + \\ + \underbrace{\left(\left(A_0 + A_1\right)\left(B_0 + B_1\right) - A_1 B_1 - A_0 B_0\right)}_{=CS_1} \cdot 2^m + \underbrace{A_0 B_0}_{=CS_0} \cdot 2^0 \end{aligned}$$

(30)

Hier werden nur 6 verschiedene Teil-Produkte im Vergleich zu 9 bei der klassischen Multiplikations-Methode (siehe **(29)**) berechnet. Es werden aber mehr Additionen benötigt. Ähnlich wie in **(28)** wurden hier die Paare von Teil-Produkten aus **(29)** durch die anderen Teil-Produkte folgendermaßen ersetzt:

$$\begin{aligned}(A_2B_1 + A_1B_2) &= (A_2 + A_1)(B_2 + B_1) - A_1B_1 - A_2B_2\\ (A_2B_0 + A_0B_2) &= (A_2 + A_0)(B_2 + B_0) - A_0B_0 - A_2B_2\\ (A_1B_0 + A_0B_1) &= (A_0 + A_1)(B_0 + B_1) - A_1B_1 - A_0B_0\end{aligned}$$

(31)

Zwei weitere Multiplikations-Formeln für 3-Segment-große Operanden wurden in [17] veröffentlicht.

In [18] haben die Autoren die Karatsuba-Formel für jedes Paar der Teil-Produkte (A_iB_j+A_jB_i) folgendermaßen generalisiert:

$$(A_iB_j + A_jB_i) = \underbrace{(A_i + A_j)(B_i + B_j)}_{=D_{ij}} - \underbrace{A_iB_i}_{=D_i} - \underbrace{A_jB_j}_{=D_j} = D_{ij} - D_i - D_j$$

(32)

Bei der Zerlegung der Faktoren in *n* Segmente kann jedes Paar der Teil-Produkte (A_iB_j+A_jB_i) aus der klassischen Multiplikations-Formel **(33)** durch die anderen Teil-Produkte nach **(32)** ersetzt werden. Damit entsteht die generalisierte Karatsuba-Multiplikations-Methode der *n*-Segment-Faktoren **(34)**.

$$A \cdot B = (A_{n-1} \cdot 2^{(n-1)\cdot m} + \ldots + A_0 \cdot 2^0) \cdot (B_{n-1} \cdot 2^{(n-1)\cdot m} + \ldots + B_0 \cdot 2^0) = \sum_{i=0}^{2n-2} CS_i \cdot 2^{i\cdot m},$$

mit

$$CS_i = \sum_{\substack{n>j\geq 0,\\ n>k\geq 0,\\ j+k=i}} A_j \cdot B_k = \begin{cases} A_{\frac{i}{2}} \cdot B_{\frac{i}{2}} + \sum_{j=0}^{\frac{i}{2}-1} \left(A_j \cdot B_{i-j} + A_{i-j} \cdot B_j\right), & \text{gerades } i \mid 0 \leq i \leq n-1 \\ \sum_{j=0}^{\frac{i-1}{2}} \left(A_j \cdot B_{i-j} + A_{i-j} \cdot B_j\right), & \text{ungerades } i \mid 0 \leq i \leq n-1 \\ A_{\frac{i}{2}} \cdot B_{\frac{i}{2}} + \sum_{j=i-(n-1)}^{\frac{i}{2}-1} \left(A_j \cdot B_{i-j} + A_{i-j} \cdot B_j\right), & \text{gerades } i \mid n \leq i \leq 2n-2 \\ \sum_{j=i-(n-1)}^{\frac{i-1}{2}} \left(A_j \cdot B_{i-j} + A_{i-j} \cdot B_j\right), & \text{ungerades } i \mid n \leq i \leq 2n-2 \end{cases}$$

$$\Leftrightarrow \begin{aligned} CS_0 &= A_0B_0\\ CS_1 &= A_0B_1 + A_1B_0\\ CS_2 &= A_0B_2 + A_1B_1 + A_2B_0\\ CS_3 &= A_0B_3 + A_1B_2 + A_2B_1 + A_3B_0\\ &\ldots\\ CS_{n-1} &= A_0B_{n-1} + A_1B_{n-2} + \ldots + A_{n-1}B_0\\ CS_n &= A_1B_{n-1} + A_2B_{n-2} + \ldots + A_{n-1}B_1\\ &\ldots\\ CS_{2n-3} &= A_{n-2}B_{n-1} + A_{n-1}B_{n-2}\\ CS_{2n-2} &= A_{n-1}B_{n-1}\end{aligned}$$

(33)

Die Formel **(33)** ist eine detaillierte Darstellung der klassischen Multiplikations-Formel **(25)**. Sie hilft die generalisierte Karatsuba-Multiplikations-Formel herzuleiten und ihre Komplexität zu berechnen. Die Formel **(33)** beinhaltet n^2 TP, wobei *n* davon $A_iB_i = D_i$ sind. Das heißt, (n^2-n) TP bilden die Paare A_iB_j+A_jB_i. So kommt die Formel **(32)** (n^2-n) Mal zum Einsatz in **(33)**. Formel **(34)** zeigt das Ergebnis des Einsatzes.

$$A \cdot B = (A_{n-1} \cdot 2^{(n-1)\cdot m} + \ldots + A_0 \cdot 2^0) \cdot (B_{n-1} \cdot 2^{(n-1)\cdot m} + \ldots + B_0 \cdot 2^0) = \\ = CS_{2n-2} \cdot 2^{(2n-2)\cdot m} + CS_{2n-3} \cdot 2^{(2n-3)\cdot m} + \ldots + CS_0 \cdot 2^0 ,$$

mit

$$CS_i = \begin{cases} D_{\frac{i}{2}} + \sum_{j=0}^{\frac{i}{2}-1} \left(D_{j(i-j)} - D_j - D_{i-j}\right), \text{gerades } i \middle| 0 \le i \le n-1 \\ \sum_{j=0}^{\frac{i-1}{2}} \left(D_{j(i-j)} - D_j - D_{i-j}\right), \text{ungerades } i \middle| 0 \le i \le n-1 \\ D_{\frac{i}{2}} + \sum_{j=i-(n-1)}^{\frac{i-2}{2}} \left(D_{j(i-j)} - D_j - D_{i-j}\right), \text{gerades } i \middle| n \le i \le 2n-2 \\ \sum_{j=i-(n-1)}^{\frac{i-1}{2}} \left(D_{j(i-j)} - D_j - D_{i-j}\right), \text{ungerades } i \middle| n \le i \le 2n-2 \end{cases}$$

(34)

Jeder Einsatz der Formel **(32)** bringt folgende Komplexitäts-Änderungen:

- anstatt zweier verschiedener Teil-Produkte wird nur 1 neues TP ermittelt
- anstatt einer Addition der 2-Segment-großen Operanden werden zwei Additionen der 1-Segment-großen Operanden und zwei Subtraktionen der 2-Segment-großen Operanden durchgeführt

Auch die generalisierte Karatsuba-Multiplikations-Methode kann mehrmals rekursiv angewendet werden. Von [18] wurde die Anzahl der Multiplikationen und die Anzahl der Additionen bei der rekursiven Anwendung der generalisierten Karatsuba-Multiplikations-Methode berechnet. Nach Autoren von [18] hat die rekursive Anwendung der Methode die minimale Komplexität, wenn die Länge der Multiplikanden folgendermaßen in Prim-Faktoren zerlegt wird:

$$n = n_j \cdot n_{j-1} \cdot \ \cdots \ \cdot n_2 \cdot n_1 = \prod_{i=1}^{j} n_i \ , \qquad n_j \le n_{j-1} \le \ \cdots \ \le n_2 \le n_1 , \qquad n_i \in P$$

(35)

Formel **(36)** stellt die von den Autoren [18] berechnete Komplexität der rekursiven Anwendung des generalisierten Karatsuba-Algorithmus für Multiplikationen der *GF(p^n)*-Elemente als Anzahl der Multiplikationen und Anzahl der Additionen dar.

$$\#MUL_{\prod_{i=1}^{j} n_i} = \left(\frac{1}{2}\right)^j \prod_{i=1}^{j} n_i \left(n_i + 1\right)$$

$$\#ADD_{\prod_{i=1}^{j} n_i} = \prod_{i=2}^{j} \#MUL_{n_i} \cdot \#ADD_{n_1} + \sum_{l=2}^{j-1} \left(\prod_{i=1}^{j} \#MUL_{n_i} \cdot w_l \right) + w_j ,$$

$$\text{mit} \qquad w_l = 4 \cdot \prod_{i=1}^{l} n_i \cdot \left(n_l - 1\right) - \frac{3}{2} n_l^2 + \frac{1}{2} n_l + 1$$

(36)

Weitere Multiplikations-Formeln für die Zerlegung der Operanden in 5, 6 und 7 Segmente sind in [19] und [20] beschrieben. MM für andere Segmentierungen der Multiplikanden sind in [21], [22] und [44] veröffentlicht. So sind momentan viele Karatsuba-ähnliche Multiplikations-Methoden bekannt.

2.5.1.3. Mehraufwand-freie Karatsuba-Multiplikation

Die Karatsuba Idee für die Multiplikation der *GF(2^n)*-Elemente **(37)** kann auch folgendermaßen interpretiert werden: eine Multiplikation der m-Bits großen Operanden wird durch (*4m-1*) Booleschen XOR ersetzt.

$$\overbrace{\underbrace{A_0}_{m}\underbrace{B_1}_{m}}^{\text{1 Multiplikation}}\ \underbrace{\oplus}_{2m-1\text{ Bits}}\ \overbrace{\underbrace{A_1}_{m}\underbrace{B_0}_{m}}^{\text{1 Multiplikation}}\ \underbrace{=}_{2m-1\text{ Bits}}\ \overbrace{\underbrace{\left(\overbrace{A_0 \oplus A_1}^{m\text{ XOR-Gatter}}\right)\left(\overbrace{B_0 \oplus B_1}^{m\text{ XOR-Gatter}}\right)}_{2m-1\text{ Bits}} \oplus \underbrace{A_0 B_0}_{2m-1\text{ Bits}} \oplus \underbrace{A_1 B_1}_{2m-1\text{ Bits}}}^{\text{1 Multiplikation:}}}^{2\cdot(2m-1)\text{ XOR-Gatter}}$$

(2m-1) XOR-Gatter

(37)

Eine bekannte Tatsache ist, dass ein XOR-Gatter aus mehr Transistoren als ein AND-Gatter besteht [45]. Deswegen sind die wichtigen Hardware-Parameter eines XOR-Gatters – die Chip-Fläche und der Energieverbrauch – größer als die eines AND-Gatters. Aus diesem Grund hat die Anwendung der Karatsuba Idee nur dann einen praktischen Sinn, wenn die Bedingung **(38)** wahr ist:

$$Parameter(Teil - Mult) \quad \geq \quad (4m-1) \cdot Parameter(XOR)$$

(38)

Effiziente mehraufwandfreie Hardware- und FPGA-Implementierungen der Karatsuba-Multiplikation wurden in [26], [46]–[27] vorgestellt. Die meisten hybriden Multiplizierer verwenden die Karatsuba-Multiplikations-Formel rekursiv bis die Teil-Multiplikanden die kritische Größe *m*, ab der eine andere – z.B. klassische – MM günstig ist, erreichen. Die *m*-Bits Teil-Multiplizierer werden dann unter Verwendung klassischer MM implementiert.

Eine andere – iterative – Optimierungs-Möglichkeit besteht in der Wiederverwendung einmal berechneter Summen der Teil-Produkte. **Abbildung 4** ist die graphische Darstellung der Karatsuba MF für 2-Segment (oder 2-Term) große Polynome **(39)** und zeigt die gleichen Summanden bei den Produkt-Segmenten C_1 und C_2.

$$A \cdot B = \left(A_1 \cdot 2^m \oplus A_0\right) \cdot \left(B_1 \cdot 2^m \oplus B_0\right) =$$
$$= \underbrace{A_1 B_1}_{2m-1} \cdot 2^{2m} \oplus \underbrace{\left(\left(A_0 \oplus A_1\right)\left(B_0 \oplus B_1\right) \oplus A_0 B_0 \oplus A_1 B_1\right)}_{2m-1} \cdot 2^m \oplus \underbrace{A_0 B_0}_{2m-1} = \underbrace{C_3}_{m-1}\underbrace{C_2}_{m}\underbrace{C_1}_{m}\underbrace{C_0}_{m}$$

(39)

Das Produkt der *2m*-Bits-Polynome ist *(4m-1)*-Bits groß und besteht aus drei verschiedenen Teil-Produkten $A_0 \cdot B_0$, $A_1 \cdot B_1$ und $\left(A_0 \oplus A_1\right) \cdot \left(B_0 \oplus B_1\right)$, die *(2m-1)*-Bits groß sind. In **Abbildung 4** sind die niedrigstwertigen *m* Bits der Teil-Produkte hellgrau und die höchstwertigen *m-1* Bits dunkelgrau dargestellt.

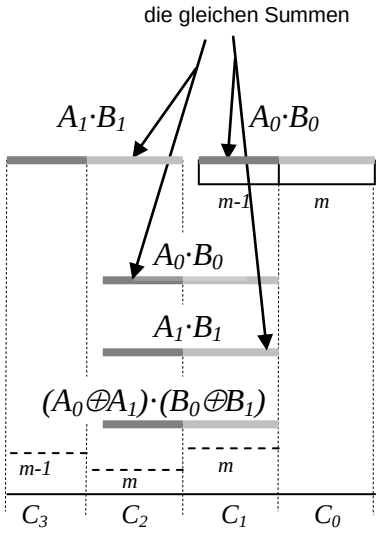


Abbildung 4: Graphische Darstellung der Karatsuba-Formel für *m*-Bits große Segmente

So ergibt sich die Komplexität des Multiplizierers der *2m*-Bits großen Polynome aus den Komplexitäten der 3 Teil-Multiplizierer (TM) und *3·(2m-1)-1* XOR Operationen. Hinzu kommen zusätzlich noch *2m* XOR Operationen, um $\left(A_0 \oplus A_1\right)$ und $\left(B_0 \oplus B_1\right)$ zu bestimmen. Das macht zusammen (*8m-4*) XOR-Gatter. Die Wiederverwendung der Summe des höchstwertigen Segmentes des $A_0 \cdot B_0$ mit dem niedrigstwertigen Segment des $A_1 \cdot B_1$ reduziert den XOR-Aufwand zu *2m+(3m-1)+2(m-1)=7m-3* XOR-Gattern. Diese Möglichkeit wurde in [27], [48] und [28] benutzt und am Beispiel der Multiplikation der 4-Segment Polynome erklärt. Auch in [49] wurde diese Idee veröffentlicht.

Die Verringerung der Anzahl der XOR-Gatter bei unveränderter Anzahl der AND-Gatter reduziert die Chip-Fläche und den Energie-Verbrauch des Multiplizierers.

Eine optimale Kombination der Multiplikations-Methoden mit dem iterativ optimierten XOR-Aufwand kann die Chip-Parameter des *GF(2^n)*-Polynom-Multiplizierers wesentlich verbessern.

Kapitel 3

Basis-Ideen der vorgeschlagenen Optimierungen

In Rahmen dieser Arbeit wurde eine Methode für die Entwicklung eines Flächen- oder/und Energie-optimierten Polynom-Multiplizierers für die Multiplikation beliebig großer Operanden entworfen. Die theoretischen Ergebnisse wurden an dem Beispiel des 233-Bits-Polynom-Multiplizierers bestätigt.

3.1. Übersicht der vorgenommenen Optimierungs-Ansätze

Abbildung 5 stellt den Zusammenhang der vorgenommenen Optimierungen dar.

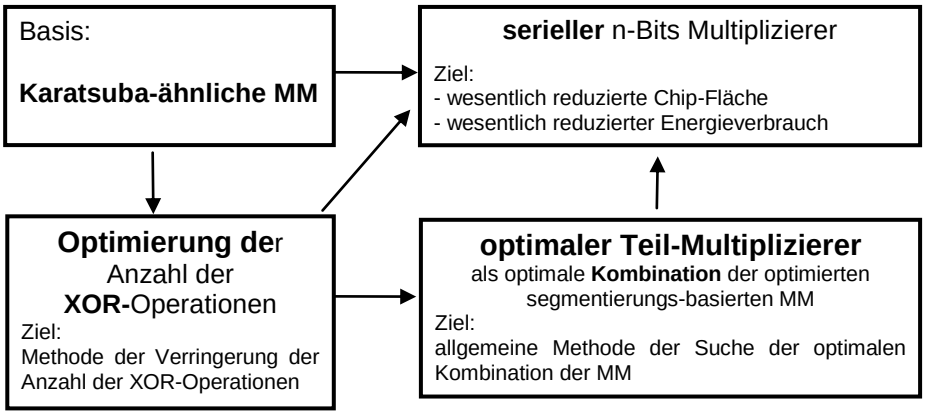


Abbildung 5: Zusammenhang der in dieser Arbeit vorgenommenen Optimierungen (Bezug zu **Abbildung 3**)

Im Gegensatz zum Stand der Technik wurden in dieser Arbeit folgende Aspekte untersucht:

- Die Entwicklung des seriellen (Mehr-Takt-) Multiplizierers nicht mehr auf Basis der klassischen MM sondern auf Basis einer Karatsuba-ähnlichen MM (siehe **Abbildung 5**). Der Vorteil ist: Karatsuba-ähnliche MM haben im Vergleich zur klassischen MM eine reduzierte Anzahl der Teil-Multiplikationen und – als Folge – eine reduzierte Anzahl der Berechnungs-Takte, was den Energie-Verbrauch der Schaltung wesentlich reduzieren kann.
- Die Entwicklung der kombinatorischen (1-Takt-) Multiplizierer auf Basis einer MM mit reduzierter Anzahl der XOR-Operationen, dass mittels optimaler Reihe der Berechnungs-Schritten (d.h. Ablaufplanes) erreicht werden kann
- Die Entwicklung des kombinatorischen (1-Takt-) Multiplizierers auf Basis einer optimalen Kombination der Multiplikations-Methoden mit der reduzierten Anzahl der XOR-Operationen.

Die Ideen, die im Rahmen dieser Arbeit für die Entwicklung eines optimalen Polynom-Multiplizierers benutzt wurden, werden in nächsten Abschnitten einzeln erklärt.

3.2. Komplexität von in Hardware implementierten Algorithmen

Die Komplexität eines Algorithmus bezieht sich auf seinen Ressourcenbedarf [51]. Der Ressourcenaufwand in Software implementierter Algorithmen ist meistens im Hinblick auf Rechenzeit und Speicherbedarf angegeben. Die Zeitkomplexität eines Algorithmus wird entweder als Anzahl aller Rechenschritte des Algorithmus angegeben oder von der Eingabe-Länge *n* abgeschätzt und mit Landau-Symbolen *O(n)* [52] angegeben. Für einen in Hardware implementierten Algorithmus sind diese Komplexitätsmaße nicht geeignet.

Bei der Hardware-Implementierung ist die minimale Chip-Fläche der Schaltung ein wichtiger Parameter, weil die Herstellungskosten davon abhängig sind. Der Speicherbedarf kann als die Anzahl der benutzten Register beschrieben werden. Die minimale Chip-Fläche, die aus den Flächen aller benutzten Logik-Gatter und Register berechnet werden kann, kann zum Vergleich verschiedener Algorithmen, die die gleiche Funktionalität bieten, benutzt werden.

Die Ausführungszeit des Algorithmus kann als die Anzahl der benötigten Takte beschrieben werden. Ein weiterer Komplexitäts-Parameter ist die Signalsverzögerungs-Zeit der Schaltung, weil sie diese maximale Taktfrequenz bestimmt.

Für mobile Geräte mit begrenzter Rechenleistung ist der Energieverbrauch eine weitere wichtige Komplexitäts-Komponente, wenn nicht sogar die wichtigste, weil die Laufzeit der mobilen Geräte von der Akku-Kapazität abhängig ist.

So kann die Komplexität eines in Hardware implementierten und für mobile Geräte vorgesehenen Algorithmus, bezeichnet mit *CA* (engl. *C*omplexity of *A*lgorithms), als eine Menge der kritischen Parameter - Ausführungszeit des Algorithmus *T* (engl. *T*ime), Energieverbrauch der Schaltung *E* (engl. *E*nergy consumption), und Chip-Fläche *A* (engl. *A*rea) - dargestellt werden:

$$CA(\text{Algorithm},\text{Technology}) = \{\ T,\ \ E,\ \ A\ \} \tag{41}$$

Jetzt wird jeder Parameter aus **(41)** einzeln analysiert, um die Komplexität ausschließlich technologie- bzw. algorithmusabhängig darzustellen.

1. Chip-Fläche *A*

Wenn die Anzahl der verwendeten Gatter je Gatter-Art bekannt ist, kann die minimale Chip-Fläche ohne Berücksichtigung der Verdrahtung zwischen den Gattern wie folgt ermittelt werden:

$$A = \#AND \cdot A_{AND} + \#XOR \cdot A_{XOR} + \ldots + \#FF \cdot A_{FF} \tag{42}$$

Hier sind:

- #*AND* - Anzahl der AND-Gatter, A_{AND} – die Fläche eines AND-Gatter
- #XOR – Anzahl der XOR-Gatter, A_{XOR} – die Fläche eines XOR-Gatter
- …
- #FF – Anzahl der Flip-Flops, A_{FF} – die Fläche eines Flip-Flops (FF)

Die Einführung der Gatter-Komplexität-Matrix *GCM* (engl. ***G***ate ***C***omplexity ***M***atrix) und der Gatter-Flächen-Matrix *GAM* (engl. ***G***ate ***A***rea ***M***atrix) erlaubt die Formel **(42)** als Matrizenmultiplikation folgendermaßen darzustellen:

$$A = GCM \cdot GAM, \qquad \text{mit } GCM = \begin{pmatrix} \#AND & \#XOR & \ldots & \#FF \end{pmatrix}$$

$$GAM = \begin{pmatrix} A_{AND} \\ A_{XOR} \\ \ldots \\ A_{FF} \end{pmatrix} \tag{43}$$

2. Energieaufwand *E*

Der dynamische Energieverbrauch hängt linear von der Anzahl der je Takt schaltenden Transistoren der Gatter ab. Jede Art der Gatter hat eine eigene bestimmte Anzahl von Transistoren und den daraus resultierenden

Energieverbrauch. Unter der Vermutung, dass der Algorithmus für *cl* Takte ausgeführt wird, kann der Energieverbrauch wie folgt berechnet werden:

$$E = cl \cdot \left(\#AND \cdot E_{AND} + \#XOR \cdot E_{XOR} + \ldots + \#FF \cdot E_{FF}\right)$$

(44)

Ähnlich wie bei der Flächen-Komplexitäts-Komponente *A* kann die Energie *E* über die Gatter-Komplexitäts-Matrix *GCM* (siehe **(43)**) und die technologieabhängige Gatter-Energieverbrauchs-Matrix (engl. ***G***ate ***E***nergie consumption ***M***atrix, bez. *GEM*) beschrieben werden. Wenn die Gatter-Energieverbrauchs-Matrix *GEM* die Energieverbrauchswerte für 1 Takt jeder Gatter-Art beinhaltet, ist der abgeschätzte Energieverbrauch wie folgt:

$$E = \underbrace{cl}_{\text{Zahl}} \cdot \underbrace{\left(GCM \cdot GEM\right)}_{\text{Matrixprodukt}}, \text{ mit } GEM = \begin{pmatrix} E_{AND} \\ E_{XOR} \\ \ldots \\ E_{FF} \end{pmatrix}$$

(45)

3. Ausführungszeit des Algorithmus *T*

Wenn der Algorithmus für *cl* Takte mit der Taktfrequenz *F* ausgeführt wird, kann die Ausführungszeit folgendermaßen berechnet werden: $T = cl \cdot \frac{1}{F}$.

Die Signalverzögerung *delay* des längsten Pfades der Schaltung (engl. *L*ongest *P*ath, bezeichnet mit *LP*) bestimmt die maximale Taktfrequenz:

$$F_{\max} = \frac{1}{delay}$$

(46)

Die Signalverzögerung besteht aus Signalverzögerungen der Gatter, die zum längsten Pfad gehören. Die Anzahl der Gatter, aus denen der LP besteht, wird durch den Index *LP* gekennzeichnet. So ist die Signalverzögerung:

$$delay = \#AND_{LP} \cdot T_{AND} + \#XOR_{LP} \cdot T_{XOR} + \ldots + \#FF_{FF} \cdot T_{FF}$$

(47)

Mit der Einführung der Gatter-Komplexitäts-Matrix des längsten Pfades GCM_{LP} und der Gatter-Zeitverzögerungs-Matrix *GTM* kann die minimale Ausführungszeit des Algorithmus als Matrizen-Produkt folgendermaßen dargestellt werden:

$$T_{\min} = cl \cdot delay = cl \cdot \underbrace{\left(GCM_{LP} \cdot GTM\right)}_{\text{Matrixprodukt}},$$

mit $$GCM_{LP} = \left(\#AND_{LP} \quad \#XOR_{LP} \quad \ldots \quad \#FF_{LP}\right)$$

$$GTM = \begin{pmatrix} T_{AND} \\ T_{XOR} \\ \dots \\ T_{FF} \end{pmatrix}$$

(48)

Aus den Formeln **(41)**, **(43)** , **(45)** und **(48)** ergibt sich die detaillierte Komplexität der Hardware-Implementierung eines beliebigen Algorithmus:

$$CA(\text{Algorithm},\text{Technology}) = \{\ T,\ E,\ A\ \} = \\ = \left\{ \underbrace{GCM_{LP}, GTM, cl}_{T},\ \underbrace{GCM, GEM, cl}_{E},\ \underbrace{GCM, GAM}_{A} \right\}$$

(49)

Die Gatter-Komplexitäts-Matrizen *GCM* und GCM_{LP} sowie auch die Anzahl der Takte *cl* sind vom gewählten Algorithmus abhängig. Die Gatter-Fläche-, der Gatter-Energieverbrauch- und die Gatter-Zeitverzögerungs-Matrizen *GAM*, *GEM* und *GTM* beinhalten nur technologieabhängige Information. Dies erlaubt die Komplexität in zwei Komponenten zu trennen:

$$CA(\text{Algorithm},\text{Technology}) = \begin{bmatrix} CA(\text{Algorithm}) = \{\ GCM_{LP},\ cl,\ GCM\ \} \\ CA(\text{Technology}) = \{\ GTM,\ GEM,\ GAM\ \} \end{bmatrix}$$

(50)

Diese Arbeit zielt auf die Optimierung der algorithmusabhängigen Komplexitäts-Komponente *CA(Algorithm)*. Für die Beschreibung der Komplexität einer Schaltung, die ein Algorithmus implementiert, wird oft die gesamte Anzahl aller Gatter dieser Schaltung benutzt. Diese Anzahl wird hier mit *#GN* bezeichnet (engl. ***G***ate ***N***umber) und als Summe aller Komponenten aus der Gatter-Komplexitäts-Matrix folgendermaßen berechnet:

$$\#GN = \#AND + \#XOR + \ldots + \#FF$$

(51)

Die Anzahl aller Gatter einer Schaltung *#GN* ist ein wichtiger Parameter der Schaltung für ihre FPGA-Implementierung. Ein FPGA (engl. ***F***ield ***P***rogrammable ***G***ate ***A***rray) besteht aus gleichartigen Komponenten, deren Funktionalität als XOR- oder AND-Gatter programmiert werden kann. Für eine ASIC-Implementierung des Algorithmus (engl. ***A***pplication-***S***pecific ***I***ntegrated ***C***ircuit) ist wichtig, wie viele Gatter je Gatter-Typ die Schaltung beinhaltet. Für die Beschreibung der Komplexität eines ASICs kann die GCM verwendet werden oder – der FPGA-Beschreibung ähnlich – die Anzahl aller Gatter des ASICs, mit der Bezeichnung des Gatter-Typs aller Summanden (z.B. als Index). Eine solche Darstellung des Inhaltes einer Schaltung wird weiterhin als ihre Gatter-Komplexität (*GC)* bezeichnet

und für die Beschreibung der Komplexität der Schaltungen der Polynom-Multiplizierer verwendet:

$$GC = \#AND_{AND} + \#XOR_{XOR} + \ldots + \#FF_{FF} \quad \textbf{(52)}$$

Z.B., wenn eine Schaltung für die Multiplikation der 5-Bits-Polynome nach der klassischen MM aus 25 AND- und 16 XOR-Gattern besteht, ist die gesamte Anzahl der Gatter dieser Schaltung *#GN=25+16=41* und die Gatter-Komplexität wie folgt: $GC=25_{AND}+16_{XOR}$.

3.3. Optimierungs-Parameter als Funktion von Gatter-Komplexität

Die Optimierung einer Schaltung kann nur im Sinn des Optimierungs-Parameters durchgeführt werden. Jeder Parameter aus **(49)** – *T*, *E* oder *A* – kann als Optimierungs-Parameter festgelegt werden. Für mobile Geräte ist entweder der Energieverbrauch *E* der Schaltung oder die Chip-Fläche *A* des Polynom-Multiplizierers als Optimierungs-Parameter sinnvoll. Die Ausführungszeit *T* für Mehr-Takt-Multiplizierer oder die Signalverzögerung *delay* für 1-Takt-Multiplizierer ist ein wichtiger zusätzlicher Parameter der Schaltung. Jeder dieser Parameter ist von der *GCM* der Schaltung abhängig:

$$T(GCM_{LP}),\quad E(GCM),\quad A(GCM) \quad \textbf{(53)}$$

Die *GCM* der Schaltung beinhaltet Gatter verschiedener Gatter-Typen. Um diese Darstellung zu erleichtern, werden Koeffizienten eingeführt, die es erlauben, einen Gatter-Typ durch einen anderen auszudrücken.

Die Polynom-Multiplikation kann mit nur zwei booleschen Funktionen – AND und XOR – implementiert werden. Aus diesem Grund kann die Gatter-Komplexität entweder als ein Tupel bei dem kombinatorischen 1-Takt-Polynom-Multiplizierer, oder als ein Tripel bei dem seriellen Mehr-Takt-Polynom-Multiplizierer wie folgt dargestellt werden:

$$GCM_{1-\text{Takt}-\text{Mult}} = (\#AND \quad \#XOR)$$
$$GCM_{\text{mehr}-\text{Takt}-\text{Mult}} = (\#AND \quad \#XOR \quad \#FF) \quad \textbf{(54)}$$

Die Technologie-Matrizen können aus der Technologie-Dokumentation ermittelt werden. Z.B. für die zwei IHP-Technologien (0,13µ und 0,25 µ) [31] erhält man folgende Matrizen:

$$GAM_{IHP(0{,}13\mu)} = \begin{pmatrix} A_{AND} = 8{,}069\,\mu m^2 \\ A_{XOR} = 13{,}45\,\mu m^2 \\ A_{FF} = 28{,}24\,\mu m^2 \end{pmatrix} \qquad GEM_{IHP(0{,}13\mu)} = \begin{pmatrix} E_{AND} = 0{,}00912\,pJ \\ E_{XOR} = 0{,}00977\,pJ \\ E_{FF} = 0{,}01429\,pJ \end{pmatrix}$$

$$GTM_{IHP(0{,}13\mu)} = \begin{pmatrix} T_{AND} = 0{,}09398ns \\ T_{XOR} = 0{,}08686ns \\ T_{FF} = 0{,}28533ns \end{pmatrix}$$

$$GAM_{IHP(0{,}25\mu)} = \begin{pmatrix} A_{AND} = A_{NAND} + A_{INV} = 42{,}336\,\mu m^2 \\ A_{XOR} = A_{XNOR} + A_{INV} = 63{,}504\,\mu m^2 \\ A_{FF} = 134{,}064\,\mu m^2 \end{pmatrix}$$

$$GEM_{IHP(0{,}25\mu)} = \begin{pmatrix} E_{AND} = 0{,}0399\,pJ \\ E_{XOR} = 0{,}0318\,pJ \\ E_{FF} = 0{,}0804\,pJ \end{pmatrix} \qquad GTM_{IHP(0{,}25\mu)} = \begin{pmatrix} T_{AND} = 0{,}31ns \\ T_{XOR} = 0{,}51ns \\ T_{FF} = 0{,}27ns \end{pmatrix}$$

(55)

IHP (0,25 µ) hat keine einzelnen AND- und XOR- Gatter. Sie wurden durch NAND-, XNOR- und INV-Gatter für theoretische Berechnungen ersetzt.

Jede Matrix von **(55)** kann wie folgt umgeformt werden:

$$GAM = \begin{pmatrix} A_{AND} \\ A_{XOR} \\ A_{FF} \end{pmatrix} = A_{AND} \cdot \begin{pmatrix} 1 \\ \dfrac{A_{XOR}}{A_{AND}} = k_A \\ \dfrac{A_{FF}}{A_{AND}} = r_A \end{pmatrix}, \qquad GEM = \begin{pmatrix} E_{AND} \\ E_{XOR} \\ E_{FF} \end{pmatrix} = E_{AND} \cdot \begin{pmatrix} 1 \\ \dfrac{E_{XOR}}{E_{AND}} = k_E \\ \dfrac{E_{FF}}{E_{AND}} = r_E \end{pmatrix}$$

(56)

Die Formeln **(43)** und **(45)** können unter Verwendung von **(54)** und **(56)** folgendermaßen umgeformt werden:

$$A = GCM \cdot GAM = GCM \cdot \left(A_{AND} \cdot \begin{pmatrix} 1 \\ k_A \\ r_A \end{pmatrix} \right) = A_{AND} \cdot \underbrace{\left(\#AND + \#XOR \cdot k_A + \#FF \cdot r_A \right)}_{\#A} =$$

$$= \#A \cdot A_{AND}$$

$$E = GCM \cdot GEM = GCM \cdot \left(E_{AND} \cdot \begin{pmatrix} 1 \\ k_E \\ r_E \end{pmatrix} \right) = E_{AND} \cdot \underbrace{\left(\#AND + \#XOR \cdot k_E + \#FF \cdot r_E \right)}_{\#E} =$$

$$= \#E \cdot E_{AND}$$

(57)

Die technologieabhängigen Koeffizienten *k* und *r* beschreiben entweder die Energieverbrauchs- oder die Flächen-Verhältnisse der Gatter-Arten. Mithilfe dieser Koeffizienten können die Optimierungs-Parameter *E* und *A* als eine äquivalente Anzahl gleichartiger Gatter #*E* und #*A* dargestellt werden. In dieser Arbeit wurden die Parameter #*E* und #*A* (siehe **(58)**) als Optimierungs-Parameter festgelegt, um den Vergleich der Komplexität verschiedener Multiplikations-Algorithmen zu erleichtern.

$$\begin{aligned} \#A &= \#AND + \#XOR \cdot k_A + \#FF \cdot r_A \\ \#E &= \#AND + \#XOR \cdot k_E + \#FF \cdot r_E \end{aligned} \qquad \textbf{(58)}$$

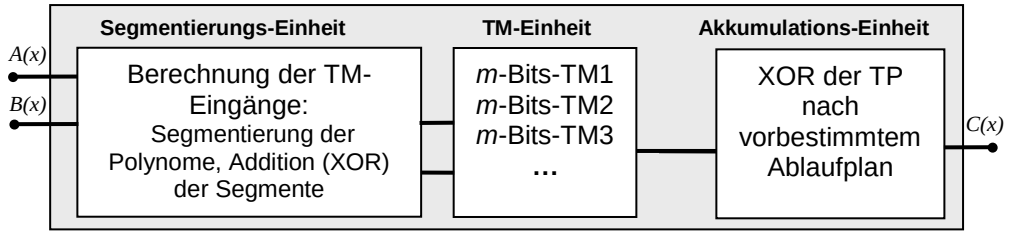
Die analytischen Ausdrücke der Optimierungs-Parameter #*E* und #*A* unterscheiden sich nur in den Koeffizienten *k* und *r* (siehe **(58)**), die die Technologie-Parameter sind. Aus diesem Grund kann die Methode der Suche nach der optimalen MM-Kombination auf beide Optimierungs-Parameter und für jede Technologie verallgemeinert werden.

Die beiden Optimierungs-Parameter in **(58)** sind Funktionen der Gatter-Komplexitäts-Matrix. Diese Parameter können nur dann berechnet werden, wenn die *GCM* oder die *GC* (siehe **(52)**), die eine praktische Darstellung der *GCM* ist, bekannt ist.

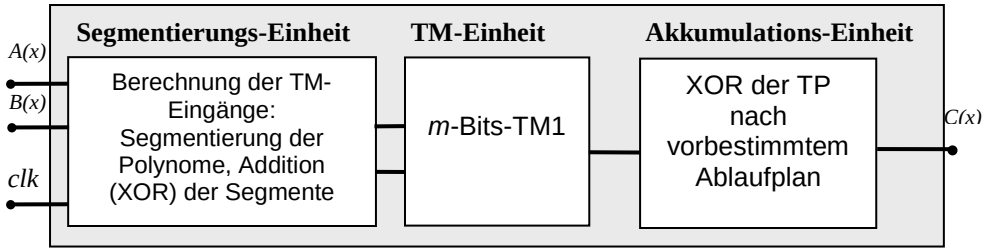
Die *GC* eines *n*-Bits-Multiplizierers kann immer als eine Funktion der *GC* der beinhalteten *m*-Bits-Teil-Multiplizierer dargestellt werden. Diese Darstellung der *GC* bietet eine Vergleichsbasis, die für die Kombination der MM verwendet werden kann.

3.4. Gatter-Komplexität als Funktion von Multiplikations-Methode und der beinhalteten Teil-Multiplizierer

Die MM großer Polynome basieren auf der Segmentierung der Faktoren und der weiteren Berechnung kleinerer Segment-Produkte. Das Produkt wird als eine Summe (XOR) dieser Segment-Produkte (Teil-Produkte) berechnet. Die digitale Schaltung eines *n*-Bits-Polynom-Multiplizierers beinhaltet die digitalen Schaltungen der *m*-Bits-Teil-Multiplizierer (*m*-Bits-TM), die die Teil-Produkte berechnen. **Abbildung 6** zeigt schematisch einen 1-Takt-Muliplizierer und einen Mehr-Takt-Multiplizierer.



a) Schematische Darstellung eines 1-Takt-Multiplizierers:
Alle Teil-Produkte werden parallel von mehreren Teil-Multipilizierern berechnet



b) Schematische Darstellung eines seriellen Mehr-Takt-Multiplizierers:
Jedes Teil-Produkt wird seriell vom gleichen Teil-Multiplizierer berechnet

Abbildung 6: Schematische Darstellung eines 1-Takt- und eines Mehr-Takt-PM

Der kombinatorische (1-Takt-) Muliplizierer beinhaltet mehrere *m*-Bits-Teil-Multiplizierer, die alle notwendigen Teil-Produkte parallel berechnen. Der serielle (Mehr-Takt-) Muliplizierer hat nur einen *m*-Bits-Teil-Multiplizierer[3], der alle notwendigen Teil-Produkte seriell (taktgesteuert) liefert. Die Anzahl der Teil-Multiplizierer #TM beim kombinatorischen Multiplizierer (bzw. die maximale Anzahl der Berechnungs-Takte clk_{max} beim seriellen Multiplizierer) ist von der gewählten Multiplikations-Methode *MM* und von der Segmentierung der Operanden n/m abhängig. Wenn die Gatter-Komplexität des *n*-Bits-Polynom-Multiplizierers mit GC_n bezeichnet wird, und die Gatter-Komplexität des *m*-Bits TM mit GC_m, kann diese Tatsache folgendermaßen dargestellt werden:

$$GC_n\left(GC_m, \#TM\left(\frac{n}{m}, MM\right)\right) \text{ für kombinatorische Multiplizierer}$$
$$GC_n\left(GC_m\right) \text{ für serielle Multiplizierer mit einem TM} \qquad \textbf{(59)}$$

Formel **(59)** gibt keine Implementierungs-Hinweise bezüglich der Teil-Multiplizierer. Diese *m*-Bits-TM können entsprechend einer beliebigen

[3] Der Mehr-Takt-Muliplizierer kann auch mehr als nur einen *m*-Bits Teil-Multiplizierer beinhalten.

Multiplikations-Methode oder nach einer beliebigen Kombination von Multiplikations-Methoden implementiert werden. Die Verwendung der gleichen Segmentierung unter Verwendung der gleichen TM ermöglicht den Vergleich der verschiedenen MM. Mithilfe **(59)** und des **Algorithmus 1**, der im nächsten Paragraph beschrieben wird, ist die komplette Analyse aller Kombinationen der MM möglich.

3.5. Algorithmische Bestimmung optimaler Kombinationen der Multiplikations-Methoden

Die Technologie-unabhängige Methode der Bestimmung der optimalen MM-Kombination für die Implementierung eines 1-Takt-Multiplizierers ist in **Algorithmus 1** beschrieben.
Die Berechnung der GC eines n-Bits-Multiplizierers bei der Segmentierung von n in n_1 Segmente kann nur dann durchgeführt werden, wenn die GC der verwendeten m-Bits-TM ($m=n/n_1$) bereits bekannt ist. Dasselbe gilt für den m-Bits-TM: die GC seiner TM müssen bereits bekannt sein. Deswegen wurde die Bestimmung der optimalen MM-Kombination für die Implementierung eines n-Bits-Multiplizierers mit der Bestimmung des optimalen 1-Bits-Multiplizierers unter Verwendung der klassischen MM begonnen und bis zu der gewünschten Länge n hochgerechnet. Wichtig ist, dass für jede kleinere Polynom-Länge i ($1<i\leq n$) alle möglichen Zerlegungen von i in n_1 Segmente durchgeführt werden. Für jede MM und für alle ihre Segmentierungen werden die Gatter-Komplexität und der Optimierungs-Parameter berechnet. Der Multiplizierer mit dem minimalen Optimierungs-Parameter wird als der optimale i-Bits-Teil-Multiplizierer weiter verwendet.
Die im **Algorithmus 1** beschriebene Methode ist nicht nur Technologie-unabhängig sondern auch für beliebig große Polynome verwendbar.
Der Schritt 2 in **Algorithmus 1** verbessert die Ergebnisse des Schrittes 1 für die Polynom-Längen, die Primzahlen sind. Diese Polynom-Länge können nicht segmentiert werden. Offensichtlich ist, dass ein n-Bits-Multiplizierer die Operanden einer kleineren Länge i, $i<n$, multiplizieren kann. Z. B. ein Multiplizierer für 168-Bits-Polynome kann immer die 163-Bits-Polynome multiplizieren. Wenn der Optimierungs-Parameter (Fläche oder Energieverbrauch) des optimierten 168-Bits-Multiplizierers kleiner als der des 163-Bits-Multiplizierers ist, wird der 168-Bits-Multiplizierer auch für die Multiplikation der 163-Bits-Polynome optimal. Schritt 2 stellt diese Optimierungs-Technik dar.

Algorithmus 1

//Bestimmung optimaler MM-Kombination für kombinatorischen Multiplizierer

Input: Länge der Polynome *n*
Optimierungs-Parameter #*A* nach **(59)** // oder #*E*
MM={ *MM_1, MM_2, MM_3, ...* }// Menge der untersuchten MM
//*MM_1* ist die aufwändigste (klassische) MM
GC als Funktion **(59)** für $\forall MM\_j \in MM$

Output: opt_GC_i

Initialisierung

$k=k_A$ // Optimierungs-Parameter ist die Chip-Fläche, sonst $k=k_E$
MM_opt[1]= *MM_1*[1] // d.h. *MM_opt*[1]= *MM_klas*[1]
for $2 \le i \le n$
MM_opt[*i*]= *empty*
end for

Berechnung

// Schritt 1: *die Bestimmung der optimalen Kombinationen von MM*
for $2 \le i \le n$ // alle kleineren Polynom-Längen
for *each* $n_1 | 2 \le n_1 \le i$, n_1 teilt *i* //alle Segmentierungen n_1 der *i*-Bits-Polynome
for *each MM_j element in MM* // alle untersuchten MM
• $GC\_temp=CG(n_1,MM\_j[i])$ //siehe **(59)**
• *opt_Param_temp= #A (GC_temp)* // siehe **(58)**
if (*MM_opt*[*i*]=*empty* or *opt_Param_temp* < *opt_Param* [*MM_opt*[*i*]])
• *MM_opt*[*i*]=*MM_j*[*i*]
end if
end for
end for
• $opt\_GC_i = GC_i(MM\_opt[i])$
end for
// Schritt 2: *Verbesserung der Ergebnisse des Schrittes 1*
for *each* $i | n > i \ge 1$
if **(***opt_Param* [*MM_opt*[*i*]] > *opt_Param* [*MM_opt*[*i+1*]]**)**
MM_opt[*i*]= *MM_opt*[i+1]
end if
end for

Die Methode der Bestimmung eines optimierten Mehr-Takt-Multiplizierers für eine vorgegebene Länge der Polynome *n* zeigt **Algorithmus 2**.

Die maximale gewünschte Ausführungszeit der Multiplikation bestimmt hier die zulässige Anzahl der Ausführungs-Takte und damit wird die maximale Anzahl der Teil-Multiplikationen vorbestimmt. Entsprechend dieser maximalen Anzahl der Teil-Produkte kann für jede MM die maximale zulässige Segmentierung der Polynome erhalten werden. Aus diesem Grund müssen die *GC* nur für zulässige Segmentierungen für jede untersuchte MM bestimmt werden.

Algorithmus 2

//Bestimmung des optimalen seriellen n-Bits-Multiplizierer

Input: Länge der Polynome *n*
Optimierungs-Parameter *#A* // oder *#E*
MM={ *MM_1, MM_2, MM_3, ...* } // Menge der untersuchten MM
GC als Funktion **(59)** für $\forall MM\_j \in MM$ // nur für die zulässigen
// Segmentierungen
$GC_i(MM\_opt[i])$ für $1 \le i \le n$ // *siehe* ***Algorithmus 1***

Output: *MM_opt*

Initialisierung
MM= { MM_1, MM_2, MM_3, ... }
MM_opt= empty

Berechnung
for *each* $n_1 | 2 \le n_1 \le n$, n_1 teilt *n* // alle Segmentierungen der *n*-Bits-Polynome
for *MM_j element in MM* // alle untersuchten Multiplikations-Methode
if **(***n_1 *is allowed for MM_j* **)** // für jede zulässige Segmentierung mit der
// *MM_j*
• $m=n/n_1$ // Länge der Segmente
• $GC\_temp=GC_n (GC_m(MM\_opt[m]), n_1, MM\_j)$ // siehe **(59)**
• *opt_Param_temp= #A (GC_temp)*
if (*MM_opt=empty* or *opt_Param_temp* < *opt_Param* [*MM_opt*])
• *MM_opt=MM_j*
end if
end if
end for
end for

Die geforderten Daten für **Algorithmus 1** und **Algorithmus 2** sind die *GC* mehrerer, am besten bereits optimierter, MM. Im Rahmen dieser Arbeit wurde eine Methode zur Reduzierung des XOR-Aufwandes bei fester Anzahl der TM vorgeschlagen. Mittels dieser Methode kann für jede MM ein optimaler Ablaufplan der Berechnung des Produktes mit minimalem XOR-Aufwand bestimmt werden. Da die Verringerung des XOR-Aufwandes infolge der *iterativen* Berechnung der Produkt-Segmente entsteht, wird die nach dem optimalen Ablaufplan durchgeführte MM weiterhin als die *iterativ optimierte* MM bezeichnet. In den nächsten Kapiteln wird diese Methode der Reduzierung des XOR-Aufwandes erklärt und die GC der *iterativ optimierten* MM wird ermittelt. Bei der Entwicklung dieser Methode wurde eine übersichtliche **T**abellarische **D**arstellung (TD) der Multiplikations-Formel verwendet.

3.6. Übersichtliche Tabellarische Darstellung einer Multiplikations-Formel

Bei großen Polynomen ist der analytische Ausdruck der Multiplikations-Formel groß, komplex und nicht mehr übersichtlich. Eine übersichtliche Darstellung hilft

jedoch die Gatter-Komplexität der Multiplikations-Methode zu berechnen. Deshalb wurde die Multiplikations-Formel in dieser Arbeit *tabellarisch* dargestellt. Mithilfe dieser übersichtlichen Darstellung des Polynom-Produktes (PP) können gleiche Operationen mit den gleichen Operanden, falls es solche gibt, leicht gefunden und eine mehrfache Berechnung vermieden werden.

Abhängig von der Größe der Segmente können zwei Arten der tabellarischen Darstellung (TD) einer MM erstellt werden:

- für die Segmentierung von *nm*-Bits-Polynomen in *n* Terme, wobei die Länge der Terme mehr als 1 Bit ist, *m>1*, bezeichnet mit MM-*n*-Segment-TD
- für die Segmentierung in *n* Terme, wo jeder Term nur 1 Bit groß ist, *m=1*, bezeichnet MM-*n*-TD.

Die Ableitung der *n*-TD aus einer *n*-Segment-TD und umgekehrt ist immer möglich.

Die Erstellung der beiden Arten der TD wird am Beispiel der Karatsuba-Multiplikations-Methode für 4-Bits (Karatsuba-4-TD) und auch für 4-Segment (Karatsuba-4-Segment-TD) große Polynome erklärt.

Formel **(60)** ist der analytische Ausdruck des Produktes der 4-Bits-Polynome, das mittels rekursiver Anwendung der Karatsuba-Multiplikations-Formel ermittelt wurde: das Produkt der 4-Bits großen Multiplikanden wurde mittels Karatsuba-Formel (siehe **(33)**) als Summe (XOR) der 3 verschiedenen Produkte der 2-Bits großen Multiplikanden dargestellt. Jedes Produkt der 2-Bits großen Multiplikanden wurde auch mittels der Karatsuba-Formel als Summe (XOR) der Produkte der 1-Bits großen Multiplikanden dargestellt.

$$A \cdot B = \underbrace{\underbrace{a_3 a_2}_{A_1}\underbrace{a_1 a_0}_{A_0}}_{A} \cdot \underbrace{\underbrace{b_3 b_2}_{B_1}\underbrace{b_1 b_0}_{B_0}}_{B} = \underbrace{a_3 a_2 \cdot b_3 b_2 \cdot 2^4}_{A_1 B_1 \cdot 2^4} \oplus$$

$$\oplus \underbrace{\left(\left(\underbrace{a_1 a_0 \oplus a_3 a_2}_{=(a_1 \oplus a_3)\cdot 2 \oplus (a_0 \oplus a_2)} \right) \cdot \left(\underbrace{b_1 b_0 \oplus b_3 b_2}_{=(b_1 \oplus b_3)\cdot 2 \oplus (b_0 \oplus b_2)} \right) \oplus a_1 a_0 \cdot b_1 b_0 \oplus a_3 a_2 \cdot b_3 b_2 \right) \cdot 2^2}_{((A_0 \oplus A_1)(B_0 \oplus B_1) \oplus A_0 B_0 \oplus A_1 B_1)\cdot 2^2} \oplus$$

$$\oplus \underbrace{a_1 a_0 \cdot b_1 b_0}_{A_0 B_0} =$$

$$= \underbrace{a_0 b_0}_{c_0} \cdot 2^0 \oplus \underbrace{\left(a_0 b_0 \oplus a_1 b_1 \oplus (a_0 \oplus a_1)(b_0 \oplus b_1)\right)}_{c_1} \cdot 2^1 \oplus$$

$$\oplus \underbrace{\left(a_0 b_0 \oplus a_1 b_1 \oplus a_2 b_2 \oplus (a_0 \oplus a_2)(b_0 \oplus b_2)\right)}_{c_2} \cdot 2^2 \oplus$$

$$\oplus \underbrace{\left(\begin{array}{l} a_0 b_0 \oplus a_1 b_1 \oplus a_2 b_2 \oplus a_3 b_3 \oplus (a_0 \oplus a_2)(b_0 \oplus b_2) \oplus \\ \oplus (a_0 \oplus a_1)(b_0 \oplus b_1) \oplus (a_1 \oplus a_3)(b_1 \oplus b_3) \oplus (a_2 \oplus a_3)(b_2 \oplus b_3) \oplus \\ \oplus (a_0 \oplus a_1 \oplus a_2 \oplus a_3)(b_0 \oplus b_1 \oplus b_2 \oplus b_3) \end{array} \right)}_{c_3} \cdot 2^3 \oplus$$

$$\oplus \underbrace{\left(a_1 b_1 \oplus a_2 b_2 \oplus a_3 b_3 \oplus (a_1 \oplus a_3)(b_1 \oplus b_3)\right)}_{c_4} \cdot 2^4 \oplus$$

$$\oplus \underbrace{\left(a_2 b_2 \oplus a_3 b_3 \oplus (a_2 \oplus a_3)(b_2 \oplus b_3)\right)}_{c_5} \cdot 2^5 \oplus \underbrace{a_3 b_3}_{c_6} \cdot 2^6 =$$

$$= c_6 \cdot 2^6 \oplus c_5 \cdot 2^5 \oplus c_4 \cdot 2^4 \oplus c_3 \cdot 2^3 \oplus c_2 \cdot 2^2 \oplus c_1 \cdot 2^1 \oplus c_0 \cdot 2^0 = c_6 c_5 c_4 c_3 c_2 c_1 c_0$$

(60)

Jedes Teil-Produkt und jeder Wert c_i in **(60)** sind 1-Bit groß. **Tabelle 1** ist die Tabellarische Darstellung der Formel **(60)**. Die linke Spalte der TD beinhaltet die Liste aller Teil-Produkte. Jede weitere Spalte der TD entspricht einem Bit des PP. Die untere Zeile der 4-TD zeigt alle diese Bits. Die Zeilen und die Spalten der TD bilden eine Art Koordinatensystem in der TD: jede TD-Zelle kann mittels *p*- und *c*-Koordinaten beschrieben werden.

Wenn das Bit c_i das TP p_j beinhaltet, wird die Zelle, die zur Zeile p_j und zur Spalte c_i gehört, mit ⊕ gefüllt: $(p_j , c_i)=\oplus$. Sonst ist diese Zelle leer: $(p_j , c_i)=0$.
Um den Wert c_i zu berechnen, müssen alle TP, deren Zellen in der Spalte c_i gefüllt sind, geXORt werden. Z.B., die Spalte c_1 hat 3 gefüllte Zellen, mit TP p_1, p_2 und p_7. Der Wert c_1 ist wie folgt: $c_1 = p_1 \oplus p_2 \oplus p_7 = a_0 \cdot b_0 \oplus a_1 \cdot b_1 \oplus (a_0 \oplus a_1) \cdot (b_0 \oplus b_1)$.

Tabelle 1: Karatsuba-4-TD

$a_0 \cdot b_0 = p_0$				$\oplus$	$\oplus$	$\oplus$	$\oplus$
$a_1 \cdot b_1 = p_1$			$\oplus$	$\oplus$	$\oplus$	$\oplus$	
$a_2 \cdot b_2 = p_2$		$\oplus$	$\oplus$	$\oplus$	$\oplus$		
$a_3 \cdot b_3 = p_3$	$\oplus$	$\oplus$	$\oplus$	$\oplus$			
$(a_0 \oplus a_2) \cdot (b_0 \oplus b_2) = p_4$				$\oplus$	$\oplus$		
$(a_1 \oplus a_3) \cdot (b_1 \oplus b_3) = p_5$			$\oplus$	$\oplus$			
$(a_0 \oplus a_1) \cdot (b_0 \oplus b_1) = p_6$				$\oplus$		$\oplus$	
$(a_2 \oplus a_3) \cdot (b_2 \oplus b_3) = p_7$		$\oplus$		$\oplus$			
$(a_0 \oplus a_1 \oplus a_2 \oplus a_3) \cdot (b_0 \oplus b_1 \oplus b_2 \oplus b_3) = p_8$				$\oplus$			
	c_6	c_5	c_4	c_3	c_2	c_1	c_0

Die Erstellung einer *n*-Segment-TD kann analog zur Erstellung der *n*-TD, mittels rekursiver Anwendung der Karatsuba MF, durchgeführt werden. Das Produkt der 4-Segment-Polynome ist in **(61)** als Summe der 1-Segment-Polynome dargestellt.

$$
\begin{aligned}
A \cdot B &= A_3A_2A_1A_0 \cdot B_3B_2B_1B_0 = \left(A_3A_2 \cdot 2^{2m} \oplus A_1A_0\right) \cdot \left(B_3B_2 \cdot 2^{2m} \oplus B_1B_0\right) = \\
&= A_3A_2 \cdot B_3B_2 \cdot 2^{4m} \oplus \left(\begin{array}{l} \left(A_1A_0 \oplus A_3A_2\right) \cdot \left(B_1B_0 \oplus B_3B_2\right) \oplus \\ \oplus\, A_1A_0 \cdot B_1B_0 \oplus A_3A_2 \cdot B_3B_2 \end{array} \right) \cdot 2^{2m} \oplus A_1A_0 \cdot B_1B_0 = \\
&= \underbrace{A_0B_0}_{CS_0} \cdot 2^0 \oplus \underbrace{\left(\begin{array}{l} A_0B_0 \oplus A_1B_1 \oplus \\ \left(A_0 \oplus A_1\right)\left(B_0 \oplus B_1\right) \end{array} \right)}_{CS_1} \cdot 2^m \oplus \underbrace{\left(\begin{array}{l} A_0B_0 \oplus A_1B_1 \oplus A_2B_2 \oplus \\ \oplus \left(A_0 \oplus A_2\right)\left(B_0 \oplus B_2\right) \end{array} \right)}_{CS_2} \cdot 2^{2m} \oplus \\
&\oplus \underbrace{\left(\begin{array}{l} A_0B_0 \oplus A_1B_1 \oplus A_2B_2 \oplus A_3B_3 \oplus \left(A_0 \oplus A_2\right)\left(B_0 \oplus B_2\right) \oplus \\ \left(A_0 \oplus A_1\right)\left(B_0 \oplus B_1\right) \oplus \left(A_1 \oplus A_3\right)\left(B_1 \oplus B_3\right) \oplus \left(A_2 \oplus A_3\right)\left(B_2 \oplus B_3\right) \oplus \\ \left(A_0 \oplus A_1 \oplus A_2 \oplus A_3\right)\left(B_0 \oplus B_1 \oplus B_2 \oplus B_3\right) \end{array} \right)}_{CS_3} \cdot 2^{3m} \oplus \\
&\oplus \underbrace{\left(A_1B_1 \oplus A_2B_2 \oplus A_3B_3 \oplus \left(A_1 \oplus A_3\right)\left(B_1 \oplus B_3\right)\right)}_{CS_4} \cdot 2^{4m} \oplus \\
&\oplus \underbrace{\left(A_2B_2 \oplus A_3B_3 \oplus \left(A_2 \oplus A_3\right)\left(B_2 \oplus B_3\right)\right)}_{CS_5} \cdot 2^{5m} \oplus \underbrace{A_3B_3}_{CS_6} \cdot 2^{6m} = \\
&= \underbrace{CS_6}_{2m-1} \cdot 2^{6m} \oplus \cdots \oplus \underbrace{CS_1}_{2m-1} \cdot 2^m \oplus \underbrace{CS_0}_{2m-1} \cdot 2^0 = \underbrace{C_7}_{m-1}\underbrace{C_6}_{m}\underbrace{C_5}_{m}\underbrace{C_4}_{m}\underbrace{C_3}_{m}\underbrace{C_2}_{m}\underbrace{C_1}_{m}\underbrace{C_0}_{m}
\end{aligned}
$$

(61)

Der Unterschied zu Formel **(60)** ist, dass hier alle TP (*2m-1*)-Bits groß sind. Deswegen können die *m*-Bits großen Segment-Werte C_i des PP nicht direkt ermittelt werden. Sie werden aus den (*2m-1*)-Bits großen Zwischenergebnissen CS_k berechnet. Dafür wird jeder Term CS_k in 2 Segmente geteilt: das niedrigstwertige *m*-Bits große Segment $CS_k[0]$ und das höchstwertige (*m-1*)-Bits große Segment $CS_k[1]$:

$$CS_k = CS_k[1] \cdot 2^m \oplus CS_k[0]$$

(62)

Die gleiche Segmentierung gilt auch für alle TP:

$$TP_j = TP_j[1] \cdot 2^m + TP_j[0]$$

(63)

Jeder Term CS_k ist eine Summe von TP. Nach der Segmentierung **(61)** und **(62)** können die Segmente des Terms CS_k als die Summe der jeweiligen TP-Segmente dargestellt werden:

$$CS_k = TP_i \oplus \ \ldots \ \oplus TP_j \qquad \Rightarrow \begin{matrix} CS_k[0] = TP_i[0] \oplus \ \ldots \ \oplus TP_j[0] \\ CS_k[1] = TP_i[1] \oplus \ \ldots \ \oplus TP_j[1] \end{matrix}$$

(64)

Jetzt können auch die Produkt-Segmente C_i von **(61)** aus CS_k-Segmenten berechnet werden:

$$C_0 = CS_0[0], \qquad C_{\underbrace{7}_{=2n-1}} = CS_{\underbrace{6}_{=2n-2}}[1], \qquad C_i = CS_i[0] \oplus CS_{i-1}[1], \ \ 0 < i < 2n-1$$

(65)

Formel **(66)** ist von **(61)** unter Verwendung von **(62)** - **(65)** abgeleitet:

$$
\begin{aligned}
A \cdot B = {} & C_7 \cdot 2^{7m} \oplus C_6 \cdot 2^{6m} \oplus C_5 \cdot 2^{5m} \oplus C_4 \cdot 2^{4m} \oplus \\
& \oplus C_3 \cdot 2^{3m} \oplus C_2 \cdot 2^{2m} \oplus C_1 \cdot 2^{m} \oplus C_0 = \underbrace{A_0 B_0[0]}_{C_0} \cdot 2^0 \oplus \\
& \oplus \underbrace{\left(A_0 B_0[0] \oplus A_0 B_0[1] \oplus A_1 B_1[0] \oplus (A_0 \oplus A_1)(B_0 \oplus B_1)[0]\right)}_{C_1} \cdot 2^m \oplus \\
& \oplus \underbrace{\begin{pmatrix} A_0 B_0[0] \oplus A_0 B_0[1] \oplus A_1 B_1[0] \oplus A_1 B_1[1] \oplus A_2 B_2[0] \oplus \\ (A_0 \oplus A_2)(B_0 \oplus B_2)[0] \oplus (A_0 \oplus A_1)(B_0 \oplus B_1)[1] \end{pmatrix}}_{C_2} \cdot 2^{2m} \oplus \\
& \oplus \underbrace{\begin{pmatrix} A_0 B_0[0] \oplus A_0 B_0[1] \oplus A_1 B_1[0] \oplus A_1 B_1[1] \oplus A_2 B_2[0] \oplus A_2 B_2[1] \oplus \\ A_3 B_3[0] \oplus (A_0 \oplus A_2)(B_0 \oplus B_2)[0] \oplus (A_0 \oplus A_2)(B_0 \oplus B_2)[1] \oplus \\ (A_1 \oplus A_3)(B_1 \oplus B_3)[0] \oplus (A_0 \oplus A_1)(B_0 \oplus B_1)[0] \oplus \\ (A_2 \oplus A_3)(B_2 \oplus B_3)[0] \oplus \\ (A_0 \oplus A_1 \oplus A_2 \oplus A_3)(B_0 \oplus B_1 \oplus B_2 \oplus B_3)[0] \end{pmatrix}}_{C_3} \cdot 2^{3m} \oplus \\
& \oplus \underbrace{\begin{pmatrix} A_0 B_0[1] \oplus A_1 B_1[0] \oplus A_1 B_1[1] \oplus A_2 B_2[0] \oplus A_2 B_2[1] \oplus A_3 B_3[0] \oplus \\ A_3 B_3[1] \oplus (A_0 \oplus A_2)(B_0 \oplus B_2)[1] \oplus (A_1 \oplus A_3)(B_1 \oplus B_3)[0] \oplus \\ (A_1 \oplus A_3)(B_1 \oplus B_3)[1] \oplus (A_0 \oplus A_1)(B_0 \oplus B_1)[1] \oplus \\ (A_2 \oplus A_3)(B_2 \oplus B_3)[1] \oplus \\ (A_0 \oplus A_1 \oplus A_2 \oplus A_3)(B_0 \oplus B_1 \oplus B_2 \oplus B_3)[1] \end{pmatrix}}_{C_4} \cdot 2^{4m} \oplus \\
& \oplus \underbrace{\begin{pmatrix} A_1 B_1[1] \oplus A_2 B_2[0] \oplus A_2 B_2[1] \oplus A_3 B_3[0] \oplus A_3 B_3[1] \oplus \\ (A_1 \oplus A_3)(B_1 \oplus B_3)[1] \oplus (A_2 \oplus A_3)(B_2 \oplus B_3)[0] \end{pmatrix}}_{C_5} \cdot 2^{5m} \oplus \\
& \oplus \underbrace{\left(A_2 B_2[1] \oplus A_3 B_3[0] \oplus A_3 B_3[1] \oplus (A_2 \oplus A_3)(B_2 \oplus B_3)[1]\right)}_{C_6} \cdot 2^{6m} \oplus \\
& \oplus \underbrace{A_3 B_3[1]}_{C_7} \cdot 2^{7m}
\end{aligned}
$$

(66)

Tabelle 2 ist die TD der Formel **(66)**. Die linke Spalte der TD beinhaltet eine Liste der Segmente aller TP. Jede weitere Spalte entspricht einem Segment des Produktes. Das Segment C_7 ist (*m-1*)-Bits groß[4]. Alle anderen Segmente des Produktes sind *m*-Bits groß. Wenn das C_i -Segment das Teil-Produkt-Segment $TP_j[d]$, $d \in \{0,1\}$ beinhaltet, wird die Zelle der TD, die zu der Zeile $TP_j[d]$ und zu der Spalte C_i gehört, mit ⊕ gefüllt: ($TP_j[d]$, C_i)=⊕. Sonst ist diese Zelle leer: ($TP_j[d]$, C_i)=0.

[4] Im Fall *n*-Segment-Polynome ist das Segment C_{2n-1} (*m-1*)-Bits groß.

Tabelle 2: Karatsuba-4-Segment-TD

$A_0 \cdot B_0[0] = TP_1[0]$					⊕	⊕	⊕	⊕
$A_0 \cdot B_0[1] = TP_1[1]$				⊕	⊕	⊕	⊕	
$A_1 \cdot B_1[0] = TP_2[0]$				⊕	⊕	⊕	⊕	
$A_1 \cdot B_1[1] = TP_2[1]$			⊕	⊕	⊕	⊕		
$A_2 \cdot B_2[0] = TP_3[0]$			⊕	⊕	⊕	⊕		
$A_2 \cdot B_2[1] = TP_3[1]$		⊕	⊕	⊕	⊕			
$A_3 \cdot B_3[0] = TP_4[0]$		⊕	⊕	⊕	⊕			
$A_3 \cdot B_3[1] = TP_4[1]$	⊕	⊕	⊕	⊕				
$(A_0 \oplus A_2) \cdot (B_0 \oplus B_2)[0] = TP_5[0]$					⊕	⊕		
$(A_0 \oplus A_2) \cdot (B_0 \oplus B_2)[1] = TP_5[1]$				⊕	⊕			
$(A_1 \oplus A_3) \cdot (B_1 \oplus B_3)[0] = TP_6[0]$				⊕	⊕			
$(A_1 \oplus A_3) \cdot (B_1 \oplus B_3)[1] = TP_6[1]$			⊕	⊕				
$(A_0 \oplus A_1) \cdot (B_0 \oplus B_1)[0] = TP_7[0]$					⊕		⊕	
$(A_0 \oplus A_1) \cdot (B_0 \oplus B_1)[1] = TP_7[1]$				⊕		⊕		
$(A_2 \oplus A_3) \cdot (B_2 \oplus B_3)[0] = TP_8[0]$			⊕		⊕			
$(A_2 \oplus A_3) \cdot (B_2 \oplus B_3)[1] = TP_8[1]$		⊕		⊕				
$(A_0 \oplus A_1 \oplus A_2 \oplus A_3) \cdot (B_0 \oplus B_1 \oplus B_2 \oplus B_3)[0] = TP_9[0]$					⊕			
$(A_0 \oplus A_1 \oplus A_2 \oplus A_3) \cdot (B_0 \oplus B_1 \oplus B_2 \oplus B_3)[1] = TP_9[1]$				⊕				
	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0

Um den Wert C_i zu berechnen, müssen die TP-Segmente, deren Zellen in der C_i-Spalte gefüllt sind, geXORt werden.

Mithilfe der TD kann die Tatsache, dass viele Segmente C_i aus den gleichen Summanden bestehen, nicht übersehen werden. Ein weiterer Vorteil der TD einer Multiplikations-Formel ist die Möglichkeit, aus einer *n*-TD die (*n-1*)-TD mit geringem Rechen- und Zeitaufwand abzuleiten. Der Grund dafür ist, dass jede *n*-Bits große Zahl mit höchstwertigem Bit 0 bereits (*n-1*)-Bits groß ist.
Im Folgenden wird die Berechnung des Produktes der Polynome nach einer MF als die Berechnung der TD dieser MF dargestellt. Die Berechnung jeder TD besteht aus den Berechnungen der:

- Multiplikanden für die Ermittlung der TP (TM-Eingänge)
- TP
- Segmente C_i (TD-Spalten)

In dieser Arbeit wird angenommen, dass die Berechnung der n-TD mit minimalem XOR-Aufwand als ein Ablaufplan, der aus der optimalen Kombination der beiden im nächsten Abschnitt eingeführten Berechnungsarten – *separat* und *iterativ* – besteht, bestimmt werden kann.

3.7. Iterative und separate Berechnung der TD-Spalten

In diesem Kapitel werden die *separate* und *iterative* Berechnungsart mathematisch beschrieben. Dafür wird jede TD-Spalte als eine Menge der TP beschrieben. Alle für n-TD eingeführten Definitionen können für n-Segment-TD erweitert werden.

Definition 1: TP-Menge P_{c_i} besteht aus allen TP der Spalte c_i:

$$P_{c_i} = \{p_k | \ (p_k, c_i) = \oplus\}, \quad 1 \le k \le \#\&, \qquad \text{mit } \#\& \text{ die Anzahl der Zeilen der TD.}$$

(67)

Definition 2: Der Wert der Spalte c_i ist die Summe (XOR) aller Elemente aus der TP-Menge P_{c_i} :

$$c_i = \bigoplus_{k=1}^{\#\&} (p_k | \ (p_k \in P_{c_i}))$$

(68)

Die Mächtigkeit der Menge $|P_{c_i}|$ zeigt die Anzahl der gefüllten Zellen in der Spalte c_i.

Definition 3: Berechnung des Wertes der Spalte c_i nach Formel **(68)** wird als ***separate*** Berechnung dieser Spalte bezeichnet.

Abbildung 7 illustriert die eingeführten Begriffe und Definitionen **(67)** und **(68)**.

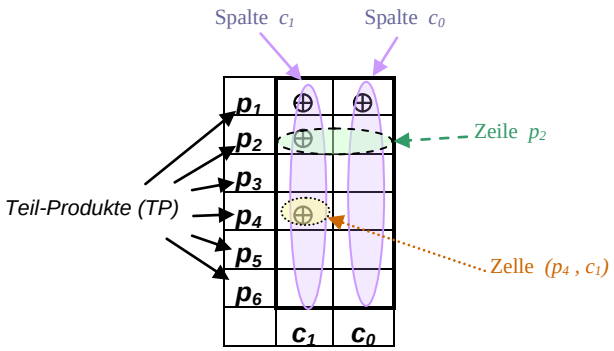


Abbildung 7: Spalte c_0 kann als TP-Menge beschrieben werden: $P_{c_0} = \{p_1\}$, $c_0 = p_1$. Spalte c_1 kann als TP-Menge beschrieben werden:

$$P_{c_1} = \{p_1, p_2, p_4\}, \ c_1 = p_1 \oplus p_2 \oplus p_4 .$$

In **Abbildung 7** ist eine TD dargestellt, die aus zwei Spalten besteht. Die Spalte c_0 hat nur eine gefüllte Zelle (p_1, c_0)=⊕. Deswegen besteht die Menge P_{c_0} aus nur

einem Teil-Produkt: $P_{c_0} = \{p_1\},\ |P_{c_0}| = 1$. Der Wert der Spalte c_0 ist dem Teil-Produkt p_1 gleich.
Die Spalte *c1* hat 3 gefüllte Zellen (p_1, c_1)=⊕, (p_2, c_1)=⊕ und (p_4, c_1)=⊕. Deswegen besteht die Menge P_{c_1} aus drei Teil-Produkten: $P_{c_1} = \{p_1, p_2, p_4\}, |P_{c_1}| = 3$. Der Wert der Spalte c_1 ist die Summe (XOR) der Teil-Produkte: $c_1 = p_1 \oplus p_2 \oplus p_4$.

Der XOR-Aufwand der *separaten* Berechnung des Wertes der Spalte c_i ist folgender:

$$\#^{separat}XOR_{c_i} = |P_{c_i}| - 1 \qquad \textbf{(69)}$$

Der XOR-Aufwand der *separaten* Berechnung aller TD-Spalten c_0 **...** c_{2n-2} kann als Summe der Komplexitäten der einzelnen Spalten folgendermaßen bestimmt werden:

$$\#^{separat}XOR_{TD-Spalten} = \sum_{i=0}^{2n-2} \#^{separat}XOR_{c_i} = \sum_{i=0}^{2n-2} \left(|P_{c_i}| - 1 \right) = \sum_{i=0}^{2n-2} |P_{c_i}| - (2n-1) \qquad \textbf{(70)}$$

Die folgenden Definitionen beziehen sich auf die Verhältnisse zweier Spalten. **Abbildung 8** illustriert diese Definitionen.

Definition 4: Wenn zwei TD-Spalten, c_i und c_j, als TP-Mengen beschrieben sind, existiert auch die Spalten-Differenzmenge der beiden. Zu der Spalten-Differenzmenge gehören die *p*-Koordinaten der Zellen, die in einer der beiden Spalten leer und in der anderen Spalte gefüllt sind:

$$P_{\Delta(c_i,c_j)} = P_{\Delta(c_j,c_i)} = P_{c_i} \Delta P_{c_j} = \left\{ p_k \middle|\ \left(p_k \in P_{c_i}\right) \wedge \left(p_k \notin P_{c_j}\right) \vee \left(p_k \notin P_{c_i}\right) \wedge \left(p_k \in P_{c_j}\right) \right\} \qquad \textbf{(71)}$$

Definition 5: Die Spalten-Differenz $\Delta(c_i,c_j)$ ist die Summe (XOR) aller Elemente aus der Menge $P_{\Delta(c_i,c_j)}$:

$$\Delta(c_i, c_j) = \overset{\#\&}{\underset{k=1}{\oplus}} \left(p_k \middle|\ p_k \in P_{\Delta(c_i,c_j)} \right) \qquad \textbf{(72)}$$

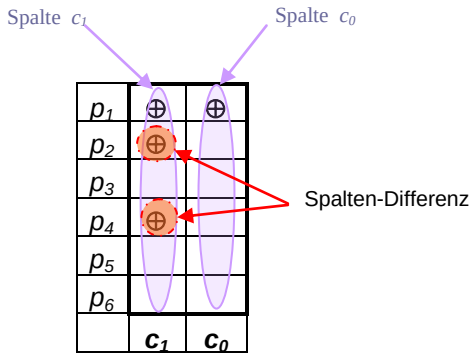


Abbildung 8: Spalten-Differenz $\Delta(c_0,c_1)$ kann als TP-Menge beschrieben werden:
$P_{\Delta(c_0,c_1)} = \{p_2, p_4\},\ \Delta(c_i, c_j) = p_2 \oplus p_4$.

Definition 6: Die Berechnung der Spalte c_i wird als *iterativ* bezeichnet, wenn sie als Summe (XOR) einer bereits berechneten Spalte c_j und der Spaltendifferenz der beiden Spalten berechnet wird:

$$c_i = c_j \oplus \Delta(c_i, c_j)$$

(73)

Aus Definition **(73)** folgt der XOR-Aufwand der *iterativen* Berechnung der Spalte c_i:

$$\#\overset{iterativ}{XOR}_{c_i} = \#XOR_{\Delta(c_i,c_j)} + 1$$

(74)

Wenn die Spalten-Differenz $\Delta(c_i,c_j)$ *separat* berechnet wurde, kann Formel **(74)** folgendermaßen detailliert geschrieben werden:

$$\#\overset{iterativ}{XOR}_{c_i} = \#\overset{separat}{XOR}_{\Delta(c_i,c_j)} + 1 = \left|P_{\Delta(c_i,c_j)}\right|$$

(75)

Die *iterative* Berechnung der Spalten-Differenz aus einer anderen, bereits ermittelten Spalten-Differenz, kann den XOR-Aufwand der Berechnung weiter verringern. **Abbildung 9** illustriert das.

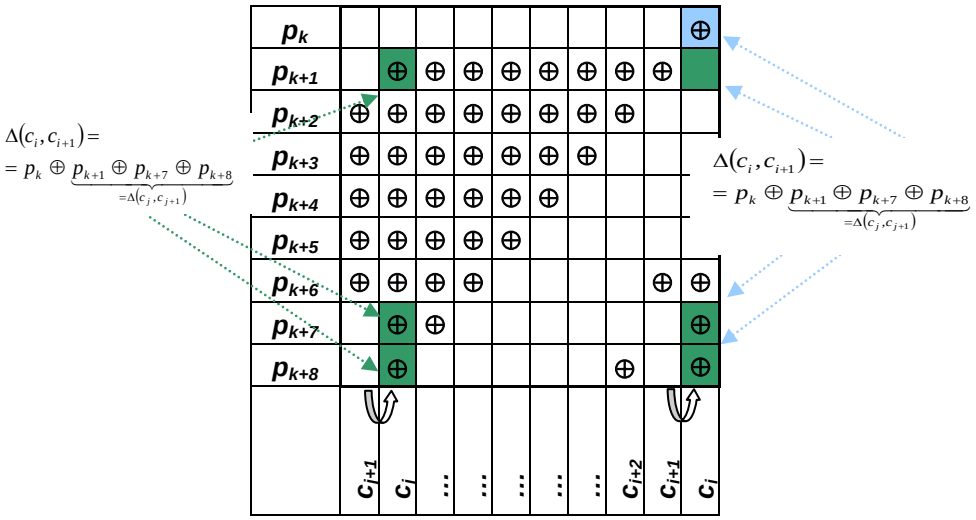


Abbildung 9: Die *iterative* Berechnung der Spalten-Differenz $\Delta(c_i, c_{i+1})$ von $\Delta(c_j, c_{j+1})$ benötigt weniger XOR-Gatter im Vergleich zu ihrer *separaten* Berechnung

Der XOR-Aufwand der *iterativen* Berechnung der Spalte c_i hängt von der Wahl der Spalte c_j ab. Wenn mehrere TD-Spalten *iterativ* berechnet werden, ist die Komplexität vom Ablaufplan der Berechnung abhängig. Dieser Zusammenhang wird bei dem XOR-Aufwand weiterhin folgendermaßen bezeichnet:

$$\#\, XOR_{\substack{berechnete\\ Spalten}}^{\text{Ablaufplan}}$$

Eine der TD-Spalten muss vorberechnet werden, um die *iterative* Berechnung der anderen TD-Spalten zu starten. Allgemein kann die Komplexität der *iterativen* TD-Berechnung folgendermaßen beschrieben werden:

$$\#\, XOR_{\substack{\text{alle Spalten}\\ \text{außer}\\ \text{Start-Spalte} c_s}}^{iterativ} = \sum_{i=0\ (i \neq s)}^{2n-2} \#\, XOR_{c_i}^{iterativ}$$

(76)

Wenn diese vorberechnete Spalte – die Start-Spalte – *separat* berechnet ist und der Ablaufplan der *iterativen* Berechnung der anderen TD-Spalten diese *separate* Vorberechnung beinhaltet, wird der Ablaufplan weiterhin als *voll iterativ* bezeichnet. Formel **(78)** beschreibt den XOR-Aufwand der *voll iterativen* Berechnung aller Spalten der TD von **Abbildung 9** nach dem Ablaufplan **(77)**.

$$\underbrace{c_{j+1}}_{\text{Start-Spalte}} \rightarrow c_j \rightarrow \ldots \rightarrow c_{i+1} \rightarrow c_i$$

(77)

$$\#\ XOR_{c_{j+1}..c_i} \overset{\substack{voll\ iterativ\\ nach\ (60)}}{=} \#XOR_{c_{j+1}}^{separat} + \sum_{l=j}^{i} \#XOR_{c_l}^{iterativ\ von\ c_{l+1}}$$

(78)

Abbildung 10 illustriert den *voll iterativen* Ablaufplan **(77)**. Die Start-Spalte c_{j+1} wird *separat* berechnet. Der XOR-Aufwand dieser Berechnung laut **(69)** beträgt $\#XOR_{c_{j+1}}^{separat} = 5-1=4$ XOR-Gatter. Alle anderen Spalten werden von links nach rechts jeweils als Summe (XOR) ihrer linken Nachbarin und der jeweiligen Spalten-Differenz ermittelt.

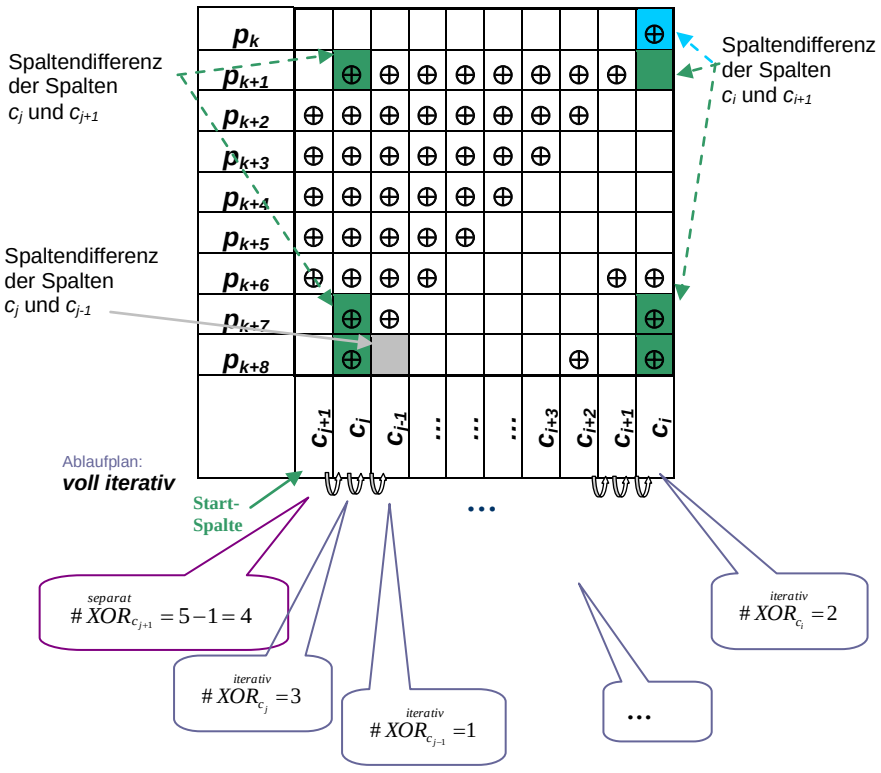


Abbildung 10: *Voll iterative* Berechnung aller TD-Spalten beginnt mit der *separaten* Berechnung der Start-Spalte c_{j+1} und der *iterativen* Berechnung der Nachbar-Spalten von links nach rechts.

Die *voll iterative* Berechnung der ganzen TD von **Abbildung 10** benötigt die folgende Anzahl XOR-Gatter, falls die Spalten-Differenz $\Delta(c_i,c_{i+1})$ aus der Spalten-Differenz $\Delta(c_j,c_{j+1})$ *iterativ* ermittelt wurde:

$$\#\ XOR_{\ c_{j+1}..c_i}^{\substack{voll\ iterativ\\ nach\ (69)}} = \underbrace{4}_{\substack{separat\\ \#XOR_{c_{j+1}}}} + \underbrace{3}_{\substack{iterativ\\ \#XOR_{c_j}}} + \underbrace{1}_{\substack{iterativ\\ \#XOR_{c_{j-1}}}} + \underbrace{1+1+1+1}_{\substack{iterativ\\ \#XOR_{c_{j-2}..c_{i+3}}}} + \underbrace{2}_{\substack{iterativ\\ \#XOR_{c_{i+2}}}} + \underbrace{3}_{\substack{iterativ\\ \#XOR_{c_{i+1}}}} + \underbrace{2}_{\substack{iterativ\\ \#XOR_{c_i}}} = 19$$

Bei der Berechnung einer TD (d.h. bei der Berechnung aller Spalten einer TD) kann jede Spalte (außer der Start-Spalte) entweder *separat* oder *iterativ* aus jeder anderen TD-Spalte berechnet werden. Der XOR-Aufwand der Berechnung aller TD-Spalten ist von der Reihenfolge der Operationen, d.h. von dem Ablaufplan, abhängig. Um einen optimalen Ablaufplan zu bestimmen, muss der XOR-Aufwand aller möglichen Ablaufpläne berechnet und verglichen werden. Diese Aufgabe kann mittels eines Graphen dargestellt werden.

3.8. Graphen-Darstellung der TD-Berechnung

Die Aufgabe alle möglichen Ablaufpläne aufzulisten und deren Komplexität (XOR-Aufwand) zu berechnen, lässt sich mittels eines Graphen (weiter TD-Graph) modellieren:

- Die Knoten des TD-Graphen werden die TD-Spalten repräsentieren
- Die Kante (c_i, c_j) zwischen zwei Knoten c_i und c_j beschreibt eine Verbindung zwischen diesen TD-Spalten durch ihre Differenzmenge, die die Anzahl der XOR-Gatter der *iterativen* Berechnung bestimmt (siehe **(74)**). So sind die Kanten für die Beschreibung der *iterativen* Berechnung vorgesehen und auf Grund von **(71)** nicht gerichtet

Ein zusätzlicher Start-Knoten c_{start} wird eingeführt, um die *separate* Berechnung jeder TD-Spalte zu modellieren: Der Knoten hat gerichtete Kanten (c_{start}, c_i). Die Richtung ist vom Knoten c_{start} zum Knoten c_i. Die Kantengewichte zeigen die Anzahl der XOR-Gatter der *separaten* Berechnung *der* Spalten c_i (siehe **(69)**). Weitere gerichtete Kanten (c_i, c_{start}) Richtung c_{start} haben ein Kantengewicht von Null.

Die Graphen-Darstellung einer TD wird am Beispiel der 3-TD der Winograd-Multiplikations-Formel erklärt.

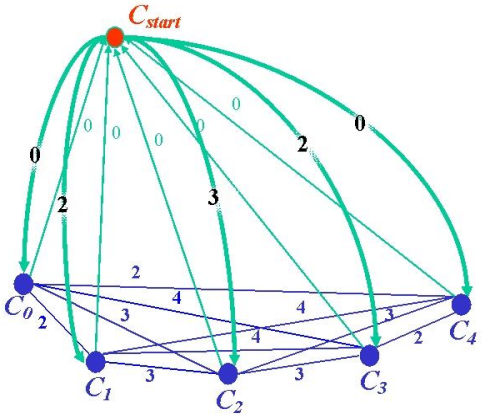
Formel **(79)** ist die analytische Darstellung des Produktes der 3-Bits-Polynome nach der Winograd-Multiplikations-Formel. **Tabelle 3** ist die TD der Formel **(79)**. **Abbildung 11** ist der Winograd-3-TD–Graph. Die Anzahl der Knoten des Graphens ist um eins größer als die Anzahl der TD-Spalten. Allgemein ist die

Anzahl der Knoten eines *n*-TD-Graphens gleich *2n* (*2n-1* Knoten für alle TD-Spalten und noch 1 Start-Knoten). Die Knoten, die den TD-Spalten entsprechen, sind blau und der zusätzliche c_{start} Knoten ist rot gekennzeichnet. Die Kanten, die die *iterative* Berechnung beschreiben, sind blau. Die gerichteten Kanten, die die *separate* Berechnung repräsentieren, sind grün und dick eingezeichnet.

$$A \cdot B = a_2a_1a_0 \cdot b_2b_1b_0 = \underbrace{a_2b_2}_{=c_4} \cdot 2^4 \oplus \underbrace{\big((a_2 \oplus a_1)(b_2 \oplus b_1) \oplus a_1b_1 \oplus a_2b_2\big)}_{=c_3} \cdot 2^3 \oplus$$
$$\oplus \underbrace{\big((a_2 \oplus a_0)(b_2 \oplus b_0) \oplus a_0b_0 \oplus a_2b_2 \oplus a_1b_1\big)}_{=c_2} \cdot 2^2 \oplus$$
$$\oplus \underbrace{\big((a_0 \oplus a_1)(b_0 \oplus b_1) \oplus a_1b_1 \oplus a_0b_0\big)}_{=c_1} \cdot 2^1 \oplus \underbrace{a_0b_0}_{=c_0} \qquad \textbf{(79)}$$

Tabelle 3: Winograd-3-TD

$a_0 \cdot b_0 = p_1$			⊕	⊕	⊕
$a_1 \cdot b_1 = p_2$		⊕	⊕	⊕	
$a_2 \cdot b_2 = p_3$	⊕	⊕	⊕		
$(a_0 \oplus a_1) \cdot (b_0 \oplus b_1) = p_4$				⊕	
$(a_0 \oplus a_2) \cdot (b_0 \oplus b_2) = p_5$			⊕		
$(a_1 \oplus a_2) \cdot (b_1 \oplus b_2) = p_6$		⊕			
	c_4	c_3	c_2	c_1	c_0



a) Tabellarische Darstellung von Formel **(79)** **b)** Winograd -3-TD-Graph

Abbildung 11: Tabellarische und Graphen-Darstellung der Winograd-Multiplikations-Formel für 3-Bits-Polynome

Nach der Einführung des Knotens C_{start} ist die Aufgabe der optimalen TD-Berechnung äquivalent zum Problem des Handelsreisenden (engl. *Traveling Salesman Problem*, weiter kurz TSP): es gibt eine bestimmte Anzahl von Städten und Abständen zwischen diesen Städten (oder Reisekosten). Der Handlungsreisende soll jede Stadt nur einmal besuchen und zurückkommen, so dass die gesamten Reisekosten minimal sind. Das Problem ist mit einem Graph modelliert. Die Knoten sind die Städte, und jede Kante zwischen zwei Knoten beschreibt eine Verbindung zwischen diesen Städten. Ziel ist es, eine möglichst kurze Tour zu finden. Der Unterschied bei der TD-Berechnung zu dem TSP ist, dass der Reisende nicht zur „Start-Stadt“ zurückkehren soll. Diese Tatsache wurde

als kostenlose Fahrt zum c_{start} Knoten modelliert, deswegen sind alle Kantengewichte Richtung c_{start} Null.

Das TSP gehört zur Klasse der NP-äquivalenten Probleme [53]. Eine exakte Lösung ist, die Reisekosten aller möglichen Rundreisen zu berechnen und eine der billigsten auszuwählen Die Anzahl der möglichen Rundreisen ist gleich der Anzahl aller Permutationen. Dies ist die Fakultät der Anzahl aller Städte (Knoten). Bezüglich der TD-Berechnung ist die Anzahl der Permutation *(2n-1)!*, weil Start- und End-Knoten fest definiert sind. Eine Rundreise eines Reisenden besteht aus *2n* Knoten entlang *2n* Kanten. Ein Kantengewicht davon ist immer Null. Wenn der XOR-Aufwand der *separaten* Berechnung eines Knotens (z.B. Knoten C_0) Null ist, werden mindestens *(2n-2)* Zahlen (Kantengewichte) pro Runde addiert. Die gesamte Anzahl der Additionen, um die Reisekosten aller möglichen Rundreisen (Permutationen) zu berechnen, ist *(2n-2)·(2n-1)!* Ein Rechner, der 1 Million Operationen pro Sekunde ausführen kann, braucht dafür $\frac{(2n-2)\cdot(2n-1)!}{10^6_{Operatios/\sec}\cdot(365\cdot 24\cdot 3660_{\sec})} = (2n-2)\cdot(2n-1)!\cdot 3{,}2\cdot 10^{-14}$ Jahre. Für die 233-Bits-Polynome dauert die Berechnung aller möglichen Ablaufpläne, die aus 465***!*** ·464 Additionen besteht, $2\cdot 10^{1029}$ Jahre.

Angenommen, mehrere Handelsreisende dürfen die Aufgabe erfüllen. Der Fall heißt multiple-TSP (kurz mTSP). Die Anzahl der Reisenden kann allgemein zwischen *1* und *2n-1* sein. Dadurch steigt die Anzahl aller möglichen Reise-Ablaufpläne rapide an.

Außerdem kann jede TD horizontal in mehrere Teile, die sich nicht überlappen, geteilt werden. **Abbildung 12** zeigt eine horizontale Teilung der TD aus **Abbildung 10** in 3 Sub-TD. In **Abbildung 12** ist jede Sub-TD mit einer anderen Farbe dargestellt.

Nach der Teilung einer TD in Sub-TD kann für jede Sub-TD ein eigener Graph gebaut werden. In diesem Fall wird der optimale Ablaufplan der Berechnung der Spalten jeder Sub-TD gesucht. Aus den ermittelten Sub-TD-Spalten werden die TD-Spalten berechnet.
Einerseits macht die Vielzahl der Sub-TD und deren Graphen die Aufgabe, die TD optimal zu berechnen, komplizierter. Anderseits kann diese Möglichkeit eine alternative Lösung werden, falls:

- die Kriterien für Teilung einer TD in Sub-TD so existieren, dass deren optimale Ablaufpläne eindeutig sind
- die Kriterien für die Ermittlung der ganzen TD-Spalten aus optimal berechneten Sub-TD-Spalten existieren

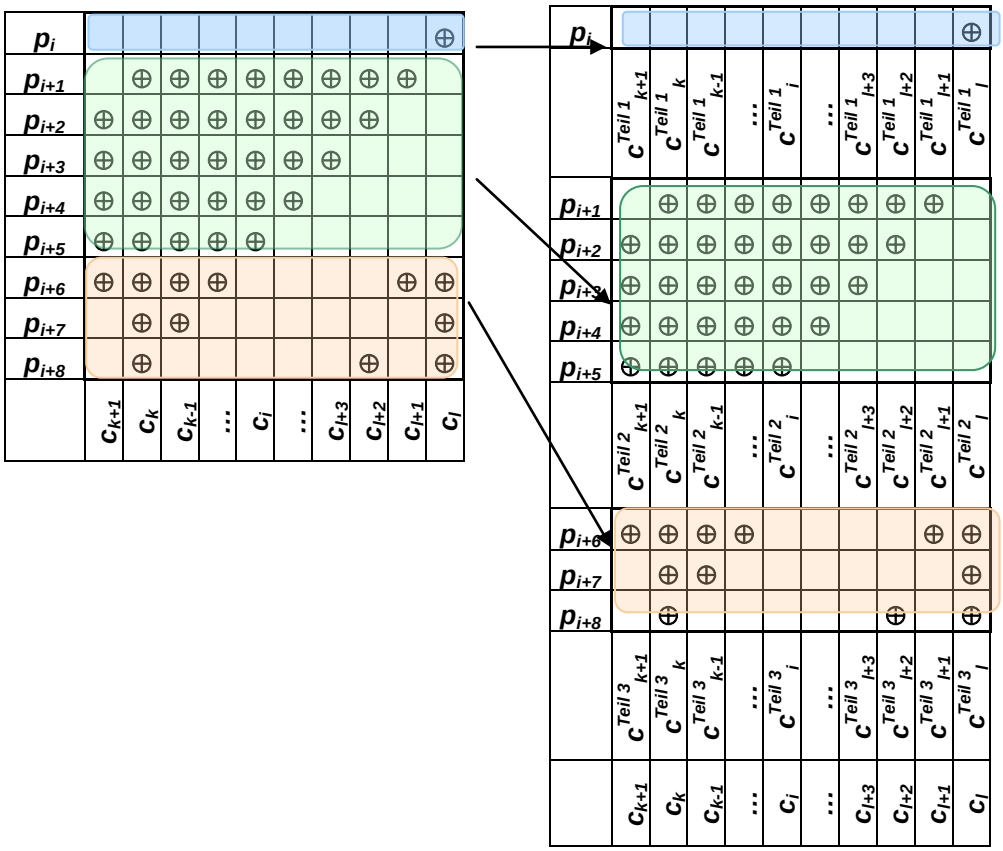


Abbildung 12: Teilung der TD **von Abbildung 10** in 3 Sub-TD

3.9. Kriterien der Teilung der TD in einen *separaten* und einen *iterativen* TD-Teil als alternative Lösung des TSP

Um die Kriterien für eine Aufteilung der TD zu ermitteln, werden die Teilung der TD in Sub-TD sowie auch die *separate* und *iterative* Berechnung der TD-Spalten aus Sub-TD-Spalten mathematisch beschrieben.

Bei horizontaler Teilung einer TD in M nicht überlappende Teile gelten für jeden m-ten Teil, $1 \le m \le M$, folgende Definitionen, die zu den Definitionen **(67)** und **(68)** mathematisch äquivalent sind:

1. Jede TP-Menge der Spalte $c_i^{Teil\ m}$ aus der Sub-TD m besteht aus p-Koordinaten gefüllter Zellen vom Teil m der ganzen Spalte c_i:

$$P_{c_i}^{\text{Teil } m} = \left\{ p_k \,\middle|\, \left(p_k, c_i^{\text{Teil } m}\right) = \oplus \right\} \tag{80}$$

2. Der Wert des *m*.Teils der Spalte c_i (bezeichnet mit $c_i^{\text{Teil } m}$) ist die Summe (XOR) aller Elemente aus $P_{c_i}^{\text{Teil } m}$:

$$c_i^{\text{Teil } m} = \bigoplus_{k=1}^{\#\&} \left(p_k \mid \left(p_k \in P_{c_i}^{\text{Teil } m} \right) \right)$$

(81)

Die Tatsache, dass die ganze TD-Spalte c_i aus M nicht überlappende Teile besteht, kann folgendermaßen beschrieben werden:

$$P_{c_i} = \bigcup_{m=1}^{M} P_{c_i}^{\text{Teil } m}, \qquad P_{c_i}^{\text{Teil } m} \cap P_{c_i}^{\text{Teil } s} = \varnothing, \quad m \neq s$$

(82)

Aus **(82)** folgt:

$$\left| P_{c_i} \right| = \sum_{m=1}^{M} \left| P_{c_i}^{\text{Teil } m} \right|$$

(83)

Definition **(68)** des ganzen Wertes c_i kann jetzt folgendermaßen für Sub-TD-Spalten-Werte beschrieben werden:

$$c_i = \bigoplus_{k=1}^{\#\&} \left(p_k \mid \left(p_k \in P_{c_i} \right) \right) = \bigoplus_{m=1}^{M} \underbrace{\bigoplus_{k=1}^{\#\&} \left(p_k \mid \left(p_k \in P_{c_i^{\text{Teil } m}} \right) \right)}_{= c_i^{m.Teil}} = \bigoplus_{m=1}^{M} c_i^{\text{Teil } m}$$

(84)

Der XOR-Aufwand der Berechnung des Wertes c_i aus allen seinen M Teil-Werten nach Formel **(84)**, also *separat*, ist folgender:

$$\#^{separat} XOR_{c_i} = \underbrace{\sum_{m=1}^{M} \#\, XOR_{c_i^{\text{Teil } m}}}_{\substack{\text{Aufwand der Berechnung} \\ \text{aller Teil-Werte der Spalte } c_i}} + \underbrace{\sum_{m=2}^{M} 1 \left| \left(c_i^{\text{Teil } m} \neq 0 \right) \right.}_{\substack{\text{Aufwand der Additionen (XOR)} \\ \text{der berechneten Teil-Werte} \\ = (M-1), \text{ wenn alle } c_i^{\text{Teil } m} \neq 0}}$$

(85)

Formel **(85)** gibt keine Hinweise bezüglich des Ablaufplanes der Berechnung jedes einzelnen *m*-ten Teils. Aufgrund **(85)** kann jede Sub-TD als eine selbstständige TD einzeln optimal, d.h. mit minimalem Aufwand, berechnet werden. Um Spalten-Werte der ganzen TD zu berechnen, werden alle berechneten Teil-Spalten-Werte (mit gleichem Index) addiert (XOR).

Die *iterative* Berechnung der Spalte c_i aus der Spalte c_j kann (analog zu **(73)**) folgendermaßen beschrieben werden:

$$c_i = \bigoplus_{m=1}^{M} c_i^{\text{Teil } m} = \bigoplus_{m=1}^{M} \left(c_j^{\text{Teil } m} \oplus \Delta\left(c_i^{\text{Teil } m}, c_j^{\text{Teil } m} \right) \right)$$

(86)

Der XOR-Aufwand der *iterativen* Berechnung **(86)** ist wie folgt:

$$\begin{aligned} \#\, XOR_{c_i}^{\text{iterativ von } c_j} &= \sum_{m=1}^{M} \#\, XOR_{c_i^{\text{Teil } m}}^{\text{iterativ von } c_j^{\text{Teil } m}} = \\ &= \sum_{m=1}^{M} \left(1 + \#\, XOR_{\Delta\left(c_i^{\text{Teil } m}, c_j^{\text{Teil } m} \right)} \right) \Big| \left(\Delta\left(c_i^{\text{Teil } m}, c_j^{\text{Teil } m} \right) \neq 0 \right) = \\ &= \sum_{m=1}^{M} \#\, XOR_{\Delta\left(c_i^{\text{Teil } m}, c_j^{\text{Teil } m} \right)} + \sum_{m=1}^{M} 1 \Big| \left(\Delta\left(c_i^{\text{Teil } m}, c_j^{\text{Teil } m} \right) \neq 0 \right) \end{aligned}$$

(87)

Jeder Wert $c_i^{\text{Teil } m}$ in **(86)** kann auch nicht nur aus der Spalte c_j, sondern auch aus jeder anderen Spalte der *m*-ten Sub-TD iterativ ermittelt werden:

$$c_i = \bigoplus_{m=1}^{M} c_i^{\text{Teil } m} = \bigoplus_{m=1}^{M} \left(c_{j|0 \le j \le 2n-2,\, j \neq i}^{\text{Teil } m} \oplus \Delta\left(c_i^{\text{Teil } m}, c_j^{\text{Teil } m} \right) \right),$$

(88)

Der XOR-Aufwand der Berechnung nach Formel **(88)** ist wie folgt:

$$\#\, XOR_{c_i}^{\text{iterativ}} = \sum_{m=1}^{M} \#\, XOR_{c_i^{\text{Teil } m}}^{\text{iterativ von } c_{j|0 \le j \le 2n-2,\, j \neq i}^{\text{Teil } m}}$$

(89)

Um die optimale Berechnung des *m*-ten Teils der TD-Spalte c_i zu bestimmen, müssen die XOR-Aufwände der *separaten* Berechnung $\#\, XOR_{c_i^{\text{Teil } m}}^{separat}$ und aller *iterativen* Berechnungen $\#\, XOR_{c_i^{\text{Teil } m}}^{\text{iterativ von } c_j^{\text{Teil } m} | \, j \neq i}$ verglichen werden.

Bei der *iterativen* Berechnung einer Reihe der TD-Spalten müssen aber alle Sub-TD nach dem gleichen Ablaufplan berechnet werden. Falls der *iterative* Ablaufplan der Berechnung eines TD-Teils optimal und eindeutig zu bestimmen ist, müssen alle anderen Sub-TD, die *iterativ* berechnet werden, auch nach diesem Ablaufplan berechnet werden. Um in einer TD einen solchen TD-Teil zu bestimmen, betrachten wir die TD als ein Set einzelner Gruppen ununterbrochen (engl. contiguous) gefüllter Zellen. Nun wird der minimale mögliche XOR-Aufwand der Berechnung einer Gruppe aus *h* vertikal ununterbrochen gefüllter Zellen diskutiert. Dann wird die Wahl der optimalen Berechnungsart – *separat* oder *iterativ* – für eine Gruppe aus *L* horizontal ununterbrochen gefüllten Zellen diskutiert und die Kriterien

für die Teilung einer TD in *separate* und *iterative* TD-Teile ausgeführt. Das alles wird an einigen abstrakten TD illustriert.

In **Tabelle 4**, die eine abstrakte TD ist, ist eine Gruppe aus *h* vertikal ununterbrochen gefüllten Zellen gezeigt. Um die eingeführten Definitionen zu benutzen, wurde die TD in Sub-TD geteilt. Die ganze Gruppe kann jetzt als eine Spalte $c_i^{\text{Teil } j}$ der Sub-TD$_j$ beschrieben werden.

Ihre *separate* Berechnung benötigt $\left|P_{c_i^{\text{Teil } j}}\right| - 1 = h - 1 = 2$ XOR-Gatter (siehe **(68)**).

Im Fall *h=1* beträgt der XOR-Aufwand 0 XOR-Gatter, was dem möglichen Minimum entspricht.

Gruppe aus *h=3* vertikal ununterbrochen gefüllten Zellen

Tabelle 4: Abstrakte TD

p_1					Sub-TD$_{i-1}$
p_2		⊕	⊕	⊕	Sub-TD$_j$
p_3		⊕	⊕	⊕	
p_4	⊕	⊕	⊕		
p_5					Sub-TD$_{j+1}$
p_6		⊕			
	c_{i+1}	c_i	c_{i-1}	c_{i-2}	

Die gewählte Gruppe kann *iterativ* aus jeder anderen Sub-TD-Spalte ermittelt werden (siehe **(74)**):

- aus der Spalte $c_{i+1}^{\text{Teil } j}$ mit 2 XOR-Gattern
- aus der Spalte $c_{i-1}^{\text{Teil } j}$ mit 0 XOR-Gattern
- aus der Spalte $c_{i-2}^{\text{Teil } j}$ mit 1 XOR-Gattern

Daraus folgt, dass der minimale Aufwand an XOR-Gattern direkt von der Spalten-Differenz der zu berechnenden Spalte mit den anderen Spalten abhängt. Bei einer Spalten-Differenz von 0 ist auch der XOR-Aufwand 0.

Es kann folgendes bewiesen werden:
Der XOR-Aufwand der *iterativen* Berechnung einer Spalte c_i von der Spalte c_j, deren TP-Mengen Pc_i und Pc_j aus der $|Pc_i|$ und $|Pc_j|$ Elementen bestehen, ist der niedrigste, wenn die Bedingung $Pc_i \subseteq Pc_j$ wahr ist:

$$\#\ XOR^{\text{iterativ von } c_j}_{c_i} = \left|P_{\Delta(c_i,c_j)}\right|_{|P_{c_j} \subseteq P_{c_i}} < \#\ XOR^{\text{iterativ von } c_j}_{c_i} = \left|P_{\Delta(c_i,c_j)}\right|_{|P_{c_j} \not\subseteq P_{c_i}}$$

(90)

Allgemein ist Folgendes zu beweisen:
Für zwei Mengen A und B mit der festen Anzahl der Elemente |A| und |B| gilt immer:

$$\left|A \Delta B\right|_{|A \subseteq B} = \min$$

Beweis:

Die Anzahl der Elemente in der Differenzmenge zweier Mengen A und B kann aus der Definition der Differenzmenge ermittelt werden:

$$\begin{aligned} A \Delta B &= A \setminus \mathrm{B} \cup \mathrm{B} \setminus \mathrm{A} \\ \left|A \Delta B\right| &= \left|A \setminus \mathrm{B}\right| + \left|\mathrm{B} \setminus \mathrm{A}\right| = \\ &= \left|A\right| - \left|A \cap B\right| + \left|B\right| - \left|B \cap A\right| = \\ &= \left|A\right| + \left|B\right| - 2 \cdot \left|A \cap B\right| \end{aligned}$$

, wobei $\left|A \cap B\right| \le \min\left(\left|A\right|, \left|B\right|\right)$

Ohne Beschränkung der Allgemeinheit gilt:

$$\left|A\right| < \left|B\right| \;\Rightarrow\; \left|A \cap B\right| \le 2 \cdot \left|A\right| \;\Rightarrow\; \left|A \Delta B\right| \;\ge\; \left|A\right| + \left|B\right| - 2 \cdot \left|A\right| = \left|B\right| - \left|A\right|$$

Die Anzahl der Elemente in der Differenzmenge zweier Mengen A und B unter der Bedingung $A \subseteq B$ ist wie folgt:

$$\begin{aligned} A \Delta B &= A \setminus \mathrm{B} \cup \mathrm{B} \setminus \mathrm{A} = \mathrm{B} \setminus \mathrm{A} \\ \left|A \Delta B\right| &= \left|\mathrm{B} \setminus \mathrm{A}\right| = \left|B\right| - \left|B \cap A\right| = \left|B\right| - \left|A\right| = \min\left(\left|A \Delta B\right|\right) \end{aligned}$$

Es kann auch gezeigt werden, dass die *iterative* Berechnung einer Spalte c_i aus der Spalte c_j optimal[5] ist, wenn c_j zwei oder mehr gefüllte Zellen hat (d.h. $\left|P_{c_j}\right|$>1) und die TP-Menge der Spalte c_j eine Untermenge der TP-Menge der Spalte c_i ist:

$$\begin{aligned} \#\,XOR_{c_i}^{\text{iterativ von } c_j} &< \#\,XOR_{c_i}^{\text{separat}} \\ \left|P_{\Delta(c_i,c_j)}\right| &< \left|P_{c_i}\right| - 1, \quad P_{c_j} \subseteq P_{c_i} \;\Rightarrow\; \left|P_{\Delta(c_i,c_j)}\right| = \left|P_{c_i}\right| - \left|P_{c_j}\right| \\ \left|P_{c_i}\right| - \left|P_{c_j}\right| &< \left|P_{c_i}\right| - 1 \\ 1 &< \left|P_{c_j}\right| \end{aligned}$$

(91)

Aus **(90)** und **(91)** folgt: Wenn für eine Reihe der TD-Spalten c_{i1}, c_{i2}, … , c_{ik} die Bedingung **(92)** gilt, ist die *iterative* Berechnung nach dem Ablaufplan **(93)** *optimal*.

$$P_{c_{i_1}} \subseteq P_{c_{i_2}} \subseteq \ldots \subseteq P_{c_{i_k}}, \quad \left|P_{c_{i_1}}\right| > 1$$

(92)

$$c_{i_1} \rightarrow c_{i_2} \rightarrow \ldots \rightarrow c_{i_k}$$

(93)

[5] d.h. den kleineren XOR-Aufwand im Vergleich zur *separaten* Berechnung hat

Falls die Bedingung **(94)** zusätzlich zur Bedingung **(92)** gilt, hat die *optimale* Berechnung nach dem Ablaufplan **(93)** den minimalen möglichen XOR-Aufwand **(95)**:

$$\left|P_{\Delta(c_{i_s},c_{i_{s-1}})}\right| \le 1 \text{ für alle } s \text{ mit } 2 \le s \le k \qquad \textbf{(94)}$$

$$\#^{optimal} XOR_{c_{i_1}\cdots c_{i_k}} = \#^{optimal} XOR_{c_{i_1}} + \sum_{s=2}^{k} 1\left(\left|P_{\Delta(c_{i_s},c_{i_{s-1}})}\right| = 1\right) \qquad \textbf{(95)}$$

Aus diesem Grund kann die Teilung einer TD in *iterative* und *separate* TD-Teile mit der Suche der Gruppe der Zellen anfangen, für welche die Bedingungen **(92)** und **(94)** wahr sind. **Tabelle 5**, die eine abstrakte TD ist, zeigt das einfachste Beispiel, in dem diese Bedingungen wahr sind.

Tabelle 5: Abstrakte TD

p_1	⊕	⊕	⊕	⊕
	c_{i+1}	c_i	c_{i-1}	c_{i-2}

Falls kein Teil einer TD die Bedingung **(92)** erfüllt, besteht die ganze TD nur aus einem *separat* zu berechnendem TD-Teil und die *separate* Berechnung ist optimal.

Ein Vergleich der beiden Berechnungsarten der abstrakten TD, die in **Tabelle 6** dargestellt ist, hilft die Kriterien für die Teilung einer TD in *iterative* und *separate* TD-Teile zu erläutern. Diese abstrakte TD besteht nur aus zwei Zeilen – p_k und p_s – aber aus mehreren Spalten. Für die Spalten wird die gewöhnliche Bezeichnung c_i weiter verwendet. Die Zeile p_k ist komplett gefüllt und die Zeile p_s hat nur L ununterbrochen gefüllten Zellen.

Tabelle 6: Abstrakte TD

p_k	⊕	⊕	…	⊕	⊕	Sub-TD 1	*iterativer* TD-Teil
p_s		⊕	…	⊕		Sub-TD 2	
	c_{i+L+1}	c_{i+L}	…	c_{i+1}	c_i		

Es wird angenommen, dass die Spalte c_i bereits bekannt ist.
Die *separate* Berechnung der Spalten c_i, c_{i+1}, ... , c_{i+L+1} benötigt die folgende Anzahl der XOR-Gatter (siehe **(70)**):

$$\#^{separat} XOR_{c_{i+1}\cdots c_{i+L+1}} = \sum_{j=i+1}^{i+L+1} \#^{separat} XOR_{c_j} = \sum_{j=i+1}^{i+L+1} \left(\left|P_{c_j}\right| - 1\right) = L \qquad \textbf{(96)}$$

Der XOR-Aufwand der *iterativen* Berechnung nach dem Ablaufplan $c_i \rightarrow c_{i+1} \rightarrow \ldots \rightarrow c_{i+L+1}$ benötigt die folgende Anzahl XOR-Gatter:

$$\#\overset{iterativ}{XOR}_{c_{i+1}\cdots c_{i+L+1}} = \sum_{j=i+1}^{i+L+1} \#\overset{iterativ\ von\ c_{j-1}}{XOR}_{c_j} = \sum_{j=i+1}^{i+L+1} \left| P_{\Delta(c_j,c_{j-1})} \right| =$$

$$= \underbrace{\left| P_{\Delta(c_i,c_{i+1})} \right|}_{=1} + 0 + \cdots + 0 + \underbrace{\left| P_{\Delta(c_{i+L},c_{i+L+1})} \right|}_{=1} = 2$$

(97)

Aus der Analyse der Formeln **(96)** und **(97)** ergeben sich die folgenden ***Kriterien (K)***, nach denen eine Gruppe, die aus *L* ununterbrochen gefüllter Zellen (p_s , c_{i+1}), **…** , (p_s, c_{i+L}) besteht, entweder zu dem *separat* oder zu dem *iterativ* zu berechnenden Sub-Teil zugeordnet wird, wenn der *iterative* Sub-Teil für die Spalten c_{i+1}, **…** , c_{i+L} bereits existiert:

1 - zum *separaten* Sub-Teil der TD, wenn *L=1*
Bemerkung: die Zelle (p_s, c_i) kann zum *iterativen* Sub-TD zugeordnet werden, wenn die *Spalte* c_i die letzte *iterativ* berechnete Spalte ist.

2 - zum *iterativen* Sub-Teil der TD, wenn *L>2* **(*K*)**

3 - entweder zum *iterativen* oder zum *separaten* Sub-Teil, wenn *L=2*
Sonderfälle: gefüllte Zellen (p_s, c_i) und (p_s, $c_{i\pm1}$) gehören zum *iterativen* Sub-TD, wenn die *Spalte* $c_{i\pm1}$ als die letzte *iterativ* (aus der *Spalte* c_i) berechnet wird.

Wenn eine TD in *iterative* und *separate* Sub-Teile zerlegt ist, besteht die Aufgabe der optimalen TD-Berechnung aus zwei unabhängigen Aufgaben:
1 – jeden Sub-Teil einzeln *optimal* berechnen
2 – alle TD-Spalten aus den jeweiligen Sub-TD-Spalten nach **(84)** zu ermitteln

Nach den ***Kriterien (K)*** können die TD verschiedener Multiplikations-Methoden in *iterative* und *separate* TD-Teile geteilt werden. Bei der klassischen Multiplikation und bei der Multiplikation nach der generalisierten Karatsuba-Formel ist der optimale Ablaufplan der Berechnung der iterativen und separaten TD-Teile eindeutig und hat den minimalen möglichen XOR-Aufwand. Für die *n*-TD der anderen MF, z.B. Karatsuba und Winograd, ist die Teilung in die *iterativen* und *separaten* TD-Teile nicht ausreichend, um den optimalen Ablaufplan eindeutig zu bestimmen. Hier müssen weitere Teilungen durchgeführt werden. Mittels einer Besonderheit der n-TD dieser MF können die Zerlegungs-Regeln eindeutig definiert werden. Die ***Kriterien (K)*** ermöglichen es, jede Sub-TD entweder dem *iterativen* oder dem *separaten* TD-Teil zuzuordnen. Die Teilung der Karatsuba- und der Winograd-*n*-TD wird im Kapitel 4 ausführlich erklärt.

Im nächsten Kapitel werden verschiedene Multiplikations-Methoden iterativ optimiert und die entsprechenden Gatter-Komplexitäten werden ermittelt.

Kapitel 4

Iterative Optimierung der Multiplikations-Methoden

In diesem Kapitel wird die Optimierung der Berechnung von Polynom-Produkten erläutert. Dabei werden 9 Multiplikations-Methoden betrachtet, beginnend mit der klassischen Multiplikations-Methode.

4.1. Klassische Multiplikations-Methode

Die klassische Multiplikations-Formel der n-Segment Polynome (siehe **(98)**) kann aus der klassischen Multiplikations-Formel der *n*-Segment-großen ganzen Zahlen (siehe **(26)** abgeleitet werden, wenn alle Additionen in **(26)** durch XOR-Operationen ersetzt werden.

$$\underbrace{C}_{2nm-1} = \underbrace{A}_{nm} \cdot \underbrace{B}_{nm} = \left(\bigoplus_{i=0}^{n-1} A_i \cdot 2^{i \cdot m} \right) \cdot \left(\bigoplus_{i=0}^{n-1} B_i \cdot 2^{i \cdot m} \right) = \bigoplus_{i=0}^{2n-2} CS_i \cdot 2^{i \cdot m}, \text{ mit } CS_i = \bigoplus_{\substack{n>j\geq 0,\\ n>k\geq 0,\\ j+k=i}} \underbrace{A_j \cdot B_k}_{2m-1} \quad \textbf{(98)}$$

Das Produkt *C* der *n·m*-Bits Polynome ist *(2nm-1)*-Bits groß und wird als die Summe (XOR) der *(2m-1)*-Bits großen Terme CS_i nach Formel **(98)** berechnet. Alle Terme CS_i sind *(2m-1)*-Bits groß und werden als Summen (XOR) der Teil-Produkte $A_i \cdot B_j$ berechnet.

Im Rahmen dieser Arbeit wird das Polynom-Produkt *C* immer als die folgende Summe (XOR) der Produkt-Segmente C_i gesucht:

$$C = \bigoplus_{i=0}^{2n-1} C_i \cdot 2^{i \cdot m} = \underbrace{C_{2n-1}}_{m-1} \underbrace{C_{2n-2}}_{m} \ldots \underbrace{C_1}_{m} \underbrace{C_0}_{m} \quad \textbf{(99)}$$

Das Produkt-Segment C_{2n-1} ist (*m-1*)-Bits groß. Alle anderen Produkt-Segmente C_i so wie auch alle Polynom-Segmente A_i, B_i sind *m*-Bits groß.

Abbildung 13 ist die graphische Darstellung der Formel **(98)** und hilft zu verstehen, wie die klass-*n*-Segment-TD und die klass-*n*-TD entstanden sind. Formel **(100)** zeigt, aus welchen TP jedes in **Abbildung 13** dargestellte Term CS_i ermittelt wird. Die niedrigstwertigen *m* Bits von jedem Teil-Produkt ($A_i \cdot B_j[0]$) sind in **Abbildung 13** hellgrau markiert, und die höchstwertigen (*m-1*) Bits, d.h. $A_i \cdot B_j[1]$, sind dunkelgrau markiert.

$$
\begin{aligned}
CS_0 &= A_0 B_0 \\
CS_1 &= A_0 B_1 \oplus A_1 B_0 \\
CS_2 &= A_0 B_2 \oplus A_1 B_1 \oplus A_2 B_0 \\
&\dots \\
CS_{n-1} &= A_0 B_{n-1} \oplus A_1 B_{n-2} \oplus \dots \oplus A_{n-1} B_0 \\
&\dots \\
CS_{2n-4} &= A_{n-3} B_{n-1} \oplus A_{n-2} B_{n-2} \oplus A_{n-1} B_{n-3} \\
CS_{2n-3} &= A_{n-2} B_{n-1} \oplus A_{n-1} B_{n-2} \\
CS_{2n-2} &= A_{n-1} B_{n-1}
\end{aligned}
\qquad \textbf{(100)}
$$

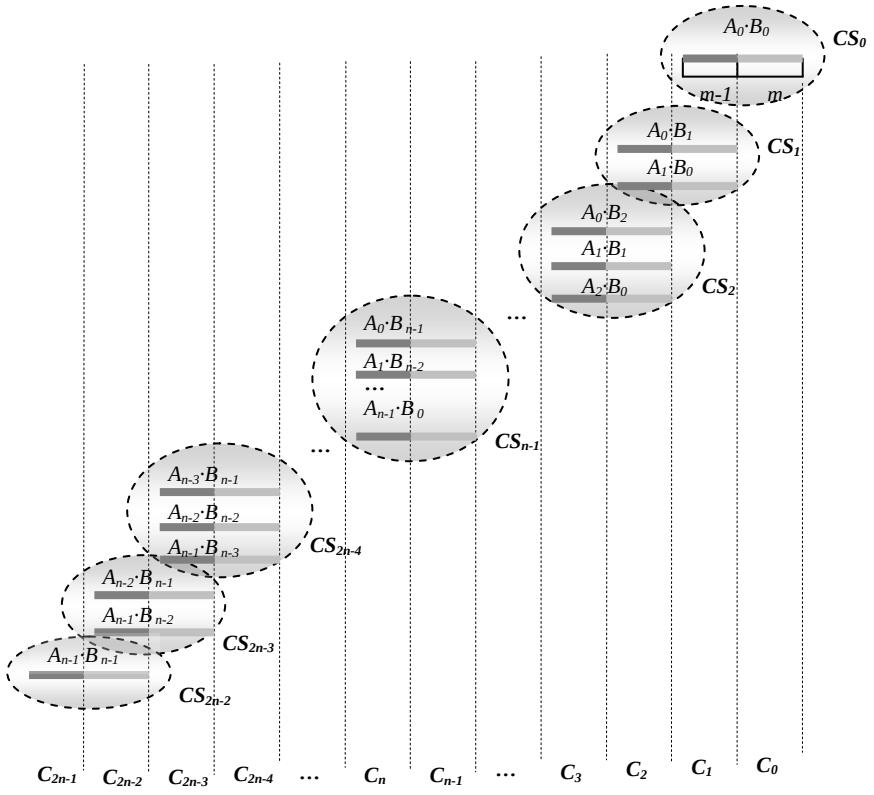


Abbildung 13: Graphische Darstellung der klassischen Multiplikations-Formel der *n*·*m*-Bits-Polynome: *n*-Segment-Polynome, *m*-Bits-Segmente

Tabelle 7 ist die TD der Formel **(98)** mit *n=4*, d.h. klass-4-Segment-TD. **Tabelle 8** ist die Tabellarische Darstellung des Sonderfalls der Formel **(98)** mit *n=4*, *m=1*, d.h. die *4*-TD. Im Fall *m=1* fehlen die höchstwertige Segmente aller TP.

Jede Zeile der klass-4-Segment-TD und der klass-4-TD (so wie auch jeder klass-*n*-Segment- oder klass-*n*-TD) hat nur eine gefüllte Zelle, d.h. für jede Zeile ist *L=1* und es gibt keine Gruppe der ununterbrochen gefüllten Zellen, für die die Bedingung **(92)** wahr ist. So gehört die ganze *n*-Segment-TD zum *separaten* Sub-Teil. Dasselbe gilt auch für *n*-TD. Deswegen ist die *iterative* Optimierung der klassischen Multiplikations-Formel der *nm*-Bits Polynome nicht möglich.

Tabelle 7: klas-4-Segment-TD

$A_0 \cdot B_0[0]$								⊕
$A_0 \cdot B_1[0]$							⊕	
$A_1 \cdot B_0[0]$							⊕	
$A_0 \cdot B_2[0]$						⊕		
$A_1 \cdot B_1[0]$						⊕		
$A_2 \cdot B_0[0]$						⊕		
$A_0 \cdot B_3[0]$					⊕			
$A_1 \cdot B_2[0]$					⊕			
$A_2 \cdot B_1[0]$					⊕			
$A_3 \cdot B_0[0]$					⊕			
$A_1 \cdot B_3[0]$				⊕				
$A_2 \cdot B_2[0]$				⊕				
$A_3 \cdot B_1[0]$				⊕				
$A_2 \cdot B_3[0]$			⊕					
$A_3 \cdot B_2[0]$			⊕					
$A_3 \cdot B_3[0]$		⊕						
$A_0 \cdot B_0[1]$							⊕	
$A_0 \cdot B_1[1]$						⊕		
$A_1 \cdot B_0[1]$						⊕		
$A_0 \cdot B_2[1]$					⊕			
$A_1 \cdot B_1[1]$					⊕			
$A_2 \cdot B_0[1]$					⊕			
$A_0 \cdot B_3[1]$				⊕				
$A_1 \cdot B_2[1]$				⊕				
$A_2 \cdot B_1[1]$				⊕				
$A_3 \cdot B_0[1]$				⊕				
$A_1 \cdot B_3[1]$			⊕					
$A_2 \cdot B_2[1]$			⊕					
$A_3 \cdot B_1[1]$			⊕					
$A_2 \cdot B_3[1]$		⊕						
$A_3 \cdot B_2[1]$		⊕						
$A_3 \cdot B_3[1]$	⊕							
	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0

Tabelle 8: klas-4-TD

$a_0 \cdot b_0$							⊕
$a_0 \cdot b_1$						⊕	
$a_1 \cdot b_0$						⊕	
$a_0 \cdot b_2$					⊕		
$a_1 \cdot b_1$					⊕		
$a_2 \cdot b_0$					⊕		
$a_0 \cdot b_3$				⊕			
$a_1 \cdot b_2$				⊕			
$a_2 \cdot b_1$				⊕			
$a_3 \cdot b_0$				⊕			
$a_1 \cdot b_3$			⊕				
$a_2 \cdot b_2$			⊕				
$a_3 \cdot b_1$			⊕				
$a_2 \cdot b_3$		⊕					
$a_3 \cdot b_2$		⊕					
$a_3 \cdot b_3$	⊕						
	c_6	c_5	c_4	c_3	c_2	c_1	c_0

Die Gatter-Komplexität der Polynom-Multiplikation kann aus **(98)** wie folgt berechnet werden:

Jeder Term CS_i (*0≤i≤n-1*) besteht aus (*i+1*) Teil-Produkten. Jeder Term CS_i (*n≤i≤2n-2*) besteht aus *(n-1)-(i-(n-1))+1=2n-i-1* Teil-Produkten. Die gesamte Anzahl der TP in **(98)** ist:

$$\#TP = \sum_{i=0}^{n-1}(i+1) + \sum_{i=n}^{2n-2}(2n-i-1) = \sum_{i=0}^{n-1} i + \sum_{i=0}^{n-1} 1 + \sum_{i=n}^{2n-2}(2n-1) - \sum_{i=n}^{2n-2} i =$$
$$= \frac{(n-1)\cdot n}{2} + n + (2n-1)\cdot(n-1) - \frac{(3n-2)\cdot(n-1)}{2} = n^2$$

(101)

Der XOR-Aufwand setzt sich zusammen aus:

- Berechnung der *(2m-1)*-Bits großen CS_i:

$$\sum_{i=0}^{n-1}\Big(\underbrace{\underbrace{\underbrace{i+1}_{\text{Anzahl der TP in } CS_i} - 1}_{\text{Anzahl der XOR-Operationen}}}\Big)\cdot(2m-1) + \sum_{i=n}^{2n-2}\Big(\underbrace{\underbrace{\underbrace{2n-i-1}_{\text{Anzahl der TP in } CS_i} - 1}_{\text{Anzahl der XOR-Operationen}}}\Big)\cdot(2m-1) =$$
$$= \left(\sum_{i=0}^{n-1} i + \sum_{i=n}^{2n-2}(2n-i-2)\right)\cdot(2m-1) = (n-1)^2\cdot(2m-1)$$

- Berechnung jedes Segmentes des Produktes C_i nach Formel **(64)**, d.h. XOR-Operation des (*m-1*)-Bits großen Segments CS_{i-1}[1] mit dem *m*-Bits großen Segment von CS_i[0], für alle *1≤i≤2n-2*: $\sum_{i=1}^{2n-2}(m-1) = (2n-2)\cdot(m-1)$

So ist die Gatter-Komplexität der klassischen Multiplikation der *n*-Segment-Polynome:

$$GC_{nm} = n^2 \cdot \overset{\text{MM des TM}}{GC}_m + (n-1)^2\cdot(2m-1)_{XOR} + (2n-2)\cdot(m-1)_{XOR} =$$
$$= n^2 \cdot \overset{\text{MM des TM}}{GC}_m + \big(2mn(n-1) - (n^2-1)\big)_{XOR}$$

(102)

Die GC des längsten Pfades, die die Signalverzögerung *delay* der klassischen MM verursacht, bezieht sich entweder auf die Berechnung der TD-Spalte C_{n-1} oder C_n. Diese beiden Spalten beinhalten je (2*n*-1) TP-Segmente. Die GC des längsten Pfades eines Teil-Multiplizierers und die Addition (XOR) von (2*n*-1) Operanden bilden die GC des längsten Pfades (LP) des klassischen Multiplizierers. Formel **(103)** stellt die GC des längsten Pfades (LP) und die verursachte Signalverzögerung dar.

$$GC_{\substack{LP\\nm}} = 1 \cdot \overset{\text{MM des } m\text{-TM}}{\text{GC}}_{LP \text{ des } m\text{-TM}} + (2n-1)_{XOR}$$

$$\overset{\text{klas}}{delay}_{nm} = \overset{\text{MM des } m\text{-TM}}{delay}_{m\text{-TM}} + \log_2(2n-1) \cdot T_{XOR}$$

(103)

Formel **(104)** entspricht der Formel **(102)** mit *m=1*. In dem Fall besteht der 1-Bits-TM aus nur 1 AND Gatter.

$$GC_m = 1_{\&} + 0_{XOR}$$
$$GC_n = n^2{}_{\&} + (n-1)^2{}_{XOR}$$
$$\overset{\text{klas}}{delay}_n = \underbrace{\overset{\text{klas MM}}{delay}_{1-PM}}_{=1 \cdot T_{AND}} + \log_2(2n-1) \cdot T_{XOR} = 1 \cdot T_{AND} + \log_2(2n-1) \cdot T_{XOR}$$

(104)

4.2. Iterative Optimierung der generalisierten Karatsuba-Multiplikation

In diesem Kapitel wird die generalisierte Karatsuba-MM [18] tabellarisch dargestellt. Die iterative Optimierung dieser MM kann mit Hilfe der TD leicht durchgeführt werden. Um die Reduzierung des XOR-Aufwandes zu demonstrieren, wird die GC der optimierten Berechnung und die GC der Berechnung aus [18] als die Funktion **(59)** dargestellt.

Die Karatsuba- und die Winograd-MM sind Sonderfälle der generalisierten Karatsuba-MM. Aus diesem Grund werden sie hier nicht einzeln untersucht, statt dessen wird ihre Gatter-Komplexität aus der GC der generalisierten Karatsuba-MM abgeleitet.

Die generalisierte Karatsuba-Multiplikations-Formel der *n*-Segment Polynome (siehe **(105)**) kann aus der generalisierten Karatsuba-Multiplikations-Formel der *n*-Segment großen ganzen Zahlen (siehe **(35)**) abgeleitet werden, wenn alle Additionen und Subtraktionen in **(35)** durch die XOR-Operationen ersetzt werden. Alle Teil-Produkte D_i und D_{ij} aus **(35)** werden hier in TP_i (bzw. TP_{ij}) umbenannt.

$$A \cdot B = (A_{n-1} \cdot 2^{(n-1)\cdot m} \oplus \ldots \oplus A_0 \cdot 2^0) \cdot (B_{n-1} \cdot 2^{(n-1)\cdot m} \oplus \ldots \oplus B_0 \cdot 2^0) = \\ = CS_{2n-2} \cdot 2^{(2n-2)\cdot m} \oplus CS_{2n-3} \cdot 2^{(2n-3)\cdot m} \oplus \ldots \oplus CS_0 \cdot 2^0 ,$$

mit

$$CS_{i|0\le i\le n-1} = \begin{cases} TP_{\frac{i}{2}} \oplus \bigoplus_{j=0}^{\frac{i}{2}-1} \left(TP_{j(i-j)} \oplus TP_j \oplus TP_{i-j}\right), & \text{gerades } i \\ \bigoplus_{j=0}^{\frac{i-1}{2}} \left(TP_{j(i-j)} \oplus TP_j \oplus TP_{i-j}\right), & \text{ungerades } i \end{cases}$$

$$CS_{i=n+k,\text{mit } 0\le k\le n-2} = \begin{cases} TP_{\frac{n+k}{2}} \oplus \bigoplus_{j=k+1}^{\frac{n+k-2}{2}} \left(TP_{j(n+k-j)} \oplus TP_j \oplus TP_{n+k-j}\right), & \text{gerades } i \\ \bigoplus_{j=k+1}^{\frac{n+k-1}{2}} \left(TP_{j(n+k-j)} \oplus TP_j \oplus TP_{n+k-j}\right), & \text{ungerades } i \end{cases}$$

$$TP_i = A_i \cdot B_i, \quad TP_{ij} = \left(A_i \oplus A_j\right) \cdot \left(B_i \oplus B_j\right)$$

(105)

Die generalisierte Karatsuba-MM ist aus der klassischen MM abgeleitet. Die Anzahl der TP in der klassischen MF ist n^2. Davon sind n Teil-Produkte $A_iB_i=TP_i$. Die restlichen (n^2-n) Teil-Produkte bilden die Paare $A_iB_j \oplus A_jB_i$, d.h. die Anzahl der Paare ist $(n^2-n)/2$. Jedes Paar ist durch die drei TP folgendermaßen ersetzbar: $A_iB_j \oplus A_jB_i = A_iB_i \oplus A_jB_j \oplus (A_i \oplus A_j)(B_i \oplus B_j) = TP_i \oplus TP_j \oplus TP_{ij}$. Die gesamte Anzahl der TP in allen CS_i ist: $n+3\cdot(n^2-n)/2=(3n^2-n)/2$. Die Anzahl der verschiedenen TP_{ij} – Teil-Produkte ist $(n^2-n)/2$ und die Anzahl der verschiedenen TP_i – Teil-Produkte ist n. D.h. nur $\#TP=n+(n^2-n)/2=(n^2+n)/2$ Teil-Produkte sind unterschiedlich.

Der XOR-Aufwand der generalisierten Karatsuba-MM [18] besteht aus:

- Berechnung der TM-Eingänge (d.h. der Operanden für die Teil-Multiplizierer, die die Teil-Produkte TP_{ij} liefern): $2m \cdot \underbrace{\frac{n^2-n}{2}}_{\text{Anzahl der TP}_{ij}} = m \cdot \left(n^2-n\right) \cdot$

- Berechnung der $(2m-1)$-Bits großen CS_i: $\left(\underbrace{\frac{3n^2-n}{2}}_{\substack{\text{Anzahl der TP} \\ \text{in allen } CS_i}} - \underbrace{(2n-1)}_{\text{Anzahl der } CS_i} \right) \cdot (2m-1)$

- Berechnung der Segmente $C_1, \ldots, C_{2n-2}$ des Produktes (siehe **(64)**):

$$\sum_{i=1}^{2n-2}(m-1)=(2n-2)\cdot(m-1)$$

Die gesamte Anzahl der XOR- Gatter ist:

$$\#XOR=\underbrace{m\cdot(n^2-n)}_{\text{Berechnung der TM-Operanden}}+\underbrace{\left(\frac{3n^2-n}{2}-(2n-1)\right)\cdot(2m-1)}_{\text{Berechnung aller } CS_i}+\underbrace{(2n-2)\cdot(m-1)}_{\text{Berechnung aller } C_i}=$$

$$=4n\cdot(n-1)\cdot m-\frac{3n^2-n}{2}+1$$

So ist die Gatter-Komplexität der *n*-Segment-PM nach der originalen generalisierten Karatsuba-Multiplikations-Methode:

$$GC_{nm}^{genKar}=\frac{n^2+n}{2}\cdot GC_{m}^{\text{MM des TM}}+\left(4n\cdot(n-1)\cdot m-\frac{3n^2-n}{2}+1\right)_{XOR}$$

(106)

Die GC des längsten Pfades, die die Signalverzögerung (*delay*) verursacht, bezieht sich auf die Berechnung entweder des Wertes C_{n-1} oder C_n. Um den Wert C_{n-1} zu erhalten, müssen die niedrigstwertigen Segmente der TP des Terms CS_{n-1} und die höchstwertigen Segmente der TP des Terms CS_{n-2} geXORt werden, d.h. ($1+3\cdot(n-1)/2+3\cdot(n-1)/2=3n-2$ Segmente der TP. Formel **(107)** stellt die GC des LP und die verursachte Signalverzögerung dar.

$$GC_{\underset{nm}{LP}}^{genKar}=1\cdot GC_{\underset{m}{LP}}^{\text{MM des TM}}+(3n-2)_{XOR}$$

$$delay_{nm}^{genKar}=delay_{m}^{\text{MM des TM}}+\log_2(3n-2)\cdot T_{XOR}$$

(107)

Um die GC der *iterativ optimierten* generalisierten Karatsuba-MM zu ermitteln, wurde die Formel **(105)** unter Verwendung von **(64)** tabellarisch dargestellt. Die *n*-TD so wie auch die *n*-Segment-TD der generalisierten Karatsuba-MF können immer eindeutig in die *iterative* und *separate* Sub-TD aufgeteilt werden. Zu der *iterativen* Sub-TD gehören die Zeilen mit den Teil-Produkten TP_i und zu der *separaten* Sub-TD mit den TP_{ij}.

Tabelle 9 zeigt das Beispiel der *n*-Segment-TD nach Formel **(105)** mit *n=4*, d.h. genKar-4-Segment-TD. **Tabelle 10** veranschaulicht den Fall der Formel **(105)** mit *m=1* und *n=4*, d.h. genKar-4-TD.

Tabelle 9: genKar-4-Segment-TD

$A_0 \cdot B_0[0]=TP_0[0]$					⊕	⊕	⊕	⊕
$A_0 \cdot B_0[1]=TP_0[1]$				⊕	⊕	⊕	⊕	
$A_1 \cdot B_1[0]=TP_1[0]$				⊕	⊕	⊕	⊕	
$A_1 \cdot B_1[1]=TP_1[1]$			⊕	⊕	⊕	⊕		
$A_2 \cdot B_2[0]=TP_2[0]$			⊕	⊕	⊕	⊕		
$A_2 \cdot B_2[1]=TP_2[1]$		⊕	⊕	⊕	⊕			
$A_3 \cdot B_3[0]=TP_3[0]$		⊕	⊕	⊕	⊕			
$A_3 \cdot B_3[1]=TP_3[1]$	⊕	⊕	⊕	⊕				
$(A_0 \oplus A_1) \cdot (B_0 \oplus B_1)$ [0] $=TP_{01}[0]$							⊕	
$(A_0 \oplus A_1) \cdot (B_0 \oplus B_1)$ [1] $=TP_{01}[1]$						⊕		
$(A_0 \oplus A_2) \cdot (B_0 \oplus B_2)$ [0]$=TP_{02}[0]$						⊕		
$(A_0 \oplus A_2) \cdot (B_0 \oplus B_2)$ [1]$=TP_{02}[1]$					⊕			
$(A_0 \oplus A_3) \cdot (B_0 \oplus B_3)$ [0]$=TP_{03}[0]$					⊕			
$(A_0 \oplus A_3) \cdot (B_0 \oplus B_3)$ [1]$=TP_{03}[1]$				⊕				
$(A_1 \oplus A_2) \cdot (B_1 \oplus B_2)$ [0]$=TP_{12}[0]$					⊕			
$(A_1 \oplus A_2) \cdot (B_1 \oplus B_2)$ [1]$=TP_{12}[1]$				⊕				
$(A_1 \oplus A_3) \cdot (B_1 \oplus B_3)$ [0]$=TP_{13}[0]$				⊕				
$(A_1 \oplus A_3) \cdot (B_1 \oplus B_3)$ [1]$=TP_{13}[1]$			⊕					
$(A_2 \oplus A_3) \cdot (B_2 \oplus B_3)$ [0]$=TP_{23}[0]$			⊕					
$(A_2 \oplus A_3) \cdot (B_2 \oplus B_3)$ [1]$=TP_{23}[1]$		⊕						
	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0

Tabelle 10: genKar-4-TD

$a_0 \cdot b_0=p_0$				⊕	⊕	⊕	⊕
$a_1 \cdot b_1=p_1$			⊕	⊕	⊕	⊕	
$a_2 \cdot b_2=p_2$		⊕	⊕	⊕	⊕		
$a_3 \cdot b_3=p_3$	⊕	⊕	⊕	⊕			
$(a_0 \oplus a_1) \cdot (b_0 \oplus b_1) = p_{01}$						⊕	
$(a_0 \oplus a_2) \cdot (b_0 \oplus b_2) = p_{02}$					⊕		
$(a_0 \oplus a_3) \cdot (b_0 \oplus b_3) = p_{03}$				⊕			
$(a_1 \oplus a_2) \cdot (b_1 \oplus b_2) = p_{12}$				⊕			
$(a_1 \oplus a_3) \cdot (b_1 \oplus b_3) = p_{13}$			⊕				
$(a_2 \oplus a_3) \cdot (b_2 \oplus b_3) = p_{23}$		⊕					
	c_6	c_5	c_4	c_3	c_2	c_1	c_0

Eine *n*-Segment-TD, mit *n>2*, kann eindeutig in *iterative* und *separate* TD-Teile aufgeteilt werden, weil:

- die TP_0[0]-Zeile den *iterativen* TD-Teil für TD-Spalten C_0-C_{n-1} bildet;
- die TP_{n-1}[1]-Zeile den *iterativen* TD-Teil für TD-Spalten C_n-C_{2n-1} bildet;
- Für alle anderen Zeilen können Kriterien ***(K)*** für eine Aufteilung angewendet werden: jede Zeile mit den TP-Segmenten TP_0[1], TP_1[0], TP_1[1],…, TP_{n-1}[0] hat *L=n* ununterbrochen gefüllte Zellen und gehört zum *iterativen* TD-Teil; der Rest gehört zum *separaten* TD-Teil.

Tabelle 11 zeigt die Aufteilung der *4*-Segment-TD (siehe **Tabelle 9**) in ihren *iterativen* und *separaten* TD-Teil, wobei bei der *iterativen* Sub-TD die gleichgefüllten Zeilen bereits zusammen geXORt wurden. Diese kompakte Darstellung wird im Folgenden als Zeilen-Implosion bezeichnet. **Tabelle 12** zeigt die Aufteilung der *4*-TD (**Tabelle 10**) in ihren *iterativen* und *separaten* TD-Teil.

Tabelle 11: Aufteilung der genKar-4-Segment-TD

Sub-TD 1: iterativer TD-Teil

$TP_0[0]$					⊕	⊕	⊕	⊕
$TP_0[1] \oplus TP_1[0]$				⊕	⊕	⊕	⊕	
$TP_1[1] \oplus TP_2[0]$			⊕	⊕	⊕	⊕		
$TP_2[1] \oplus TP_3[0]$		⊕	⊕	⊕	⊕			
$TP_3[1]$	⊕	⊕	⊕	⊕				
	C^1_7	C^1_6	C^1_5	C^1_4	C^1_3	C^1_2	C^1_1	C^1_0

Sub-TD 2: separater TD-Teil

$TP_{01}[0]$							⊕	
$TP_{01}[1]$						⊕		
$TP_{02}[0]$						⊕		
$TP_{02}[1]$					⊕			
$TP_{03}[0]$					⊕			
$TP_{03}[1]$				⊕				
$TP_{12}[0]$					⊕			
$TP_{12}[1]$				⊕				
$TP_{13}[0]$				⊕				
$TP_{13}[1]$			⊕					
$TP_{23}[0]$			⊕					
$TP_{23}[1]$		⊕						
	C^2_7	C^2_6	C^2_5	C^2_4	C^2_3	C^2_2	C^2_1	C^2_0

Tabelle 12: Aufteilung der genKar-4-TD

Sub-TD 1: iterativer TD-Teil

p_0				⊕	⊕	⊕	⊕
p_1			⊕	⊕	⊕	⊕	
p_2		⊕	⊕	⊕	⊕		
p_3	⊕	⊕	⊕	⊕			
	c^1_6	c^1_5	c^1_4	c^1_3	c^1_2	c^1_1	c^1_0

Sub-TD 2: *separater* TD-Teil

p_{01}						⊕	
p_{02}					⊕		
p_{03}				⊕			
p_{12}				⊕			
p_{13}			⊕				
p_{23}		⊕					
	c^2_6	c^2_5	c^2_4	c^2_3	c^2_2	c^2_1	c^2_0

Der XOR-Aufwand der Berechnung der *n*-Segment-TD besteht aus:

- der Berechnung der TM-Eingänge: $2m \cdot \underbrace{\frac{n^2-n}{2}}_{\text{Anzahl der } TP_{ij}} = m \cdot (n^2-n) \cdot$

- der Berechnung der Zeilen-Implosion der *iterativen* Sub-TD: $(m-1) \cdot \underbrace{(n-1)}_{\substack{\text{Anzahl der} \\ \text{Zeilen-Implosion}}}$

- der Berechnung der *iterativen* Sub-TD nach dem Ablaufplan
$C^1_{2n-1} \rightarrow C^1_{2n-2} \rightarrow \ldots \rightarrow C^1_n \quad C^1_{n-1} \leftarrow \ldots \leftarrow C^1_1 \leftarrow C^1_0$:

$$\underbrace{m \cdot (n-1) - 1}_{\text{linke } n \text{ Spalten}} + \underbrace{m \cdot (n-1)}_{\text{rechte } n \text{ Spalten}} = 2m \cdot (n-1) - 1$$

- der Addition (XOR) der TP aus der *separaten* Sub-TD : $\underbrace{\frac{n^2-n}{2}}_{\text{Anzahl der } TP_{ij}} \cdot (2m-1)$

Die gesamte Anzahl der XOR- Gatter ergibt sich aus:

$$\#XOR = \underbrace{m \cdot (n^2-n)}_{\substack{\text{Berechnung} \\ \text{TM-Operanden}}} + \underbrace{(m-1) \cdot (n-1) + 2m \cdot (n-1) - 1}_{\substack{\text{Berechnung } \textit{iterativer} \text{ Sub-TD,} \\ \text{mit der Zeilen-Implosion}}} + \underbrace{\frac{n^2-n}{2} \cdot (2m-1)}_{\substack{\text{XOR der TP von } \textit{separater} \text{ Sub-TD}}} =$$

$$= 2mn^2 + mn - 3m - \frac{n^2+n}{2} = m(2n^2+n-3) - \frac{n^2+n}{2}$$

So ist die Gatter-Komplexität der *n*-Segment PM nach der iterativ optimierten generalisierten Karatsuba-Multiplikations-Methode:

$$\overset{\text{iterativ optim}}{\underset{nm}{GC}}_{genKar} = \frac{n^2+n}{2} \cdot \overset{\text{MM des TM}}{\underset{m}{GC}} + \left(m\left(2n^2+n-3\right) - \frac{n^2+n}{2} \right)_{XOR}$$

(108)

Die GC der *iterativ* optimierten generalisierten Karatsuba-MM **(108)** kann im Vergleich zu der originalen MM **(107)** um die folgende Anzahl an XOR-Gattern verringert werden:

$$\Delta = \overset{genKar}{GC}_{nm} - \overset{\text{iterativ optim}}{\underset{nm}{GC}}_{genKar} = \left(m\left(2n^2-5n+3\right) - n^2+n+1\right)_{XOR}$$

(109)

Formel **(110)** zeigt, dass die GC für alle Segmentierungen mittels *iterativer* Optimierung reduziert werden kann.

$$\Delta = \left(2(m-1)(n-2)^2 + (n-2)^2 + 3(m-1)(n-2) + (m-1)\right)_{XOR}$$
$$\Rightarrow \Delta > 0, \quad \text{wenn } m>1,\ n \geq 2$$

(110)

Die GC des längsten Pfades bezieht sich entweder auf die Berechnung der linken *n* Spalten der TD oder der rechten *n* Spalten der TD. Die beiden Berechnungen können gleichzeitig durchgeführt werden und verursachen die gleiche Signal-Verzögerung. Die Berechnung jeder Teil-Spalte C^1_i, 1≤i≤n-1, aus dem *iterativen* TD-Teil trägt den Wert T_{XOR} zur Signal-Verzögerung des Multiplizierers bei. Die Addition (XOR) der Teil-Spalten aus der separaten Sub-TD benötigt noch eine weitere XOR-Operation. Formel **(111)** stellt die GC des LP und die verursachte Signalverzögerung (*delay*) dar.

$$\underset{nm}{GC_{LP}} = 1 \cdot \overset{\text{MM des TM}}{\underset{m}{GC_{LP}}} + \left(n+1\right)_{XOR}$$

$$\overset{genKar}{delay}_{nm} = \overset{\text{MM des TM}}{delay}_{m} + \left(n+1\right) \cdot T_{XOR}$$

(111)

4.3. Herleitung und iterative Optimierung der Karatsuba-MM für n-Term-Polynome

In diesem Kapitel wird die GK der *iterativ optimierten* Berechnung der Karatsuba-MM als eine Funktion von der Länge der Polynome *n* ermittelt. Dafür wurden die Karatsuba-n-TD, d.h. die TD der Karatsuba-MF für *n*-Bits Polynome, *1≤n≤256*, analysiert. Die exakte Gatter-Komplexität der iterativ optimierten Karatsuba-MM

kann für Karatsuba-*n*-TD für beliebigen *n* ermittelt werden. In dieser Arbeit wurden nur die ECC-relevanten Polynom-Längen *n≤600* untersucht.
Die Ermittlung der Gatter-Komplexität für n-Term Polynome, d.h. für Karatsuba-n-Segment-TD, ist eine aufwändige Aufgabe, obwohl die Ergebnisse der n-TD-Berechnung teilweise verwendet werden können. Aus diesem Grund wurde die GK für Karatsuba-*n*-Segment-TD nur für die Segmentierung der Operanden in *n* Teile, *2≤n≤16*, ermittelt. Es wurde festgestellt, dass die Segmentierung der Polynome in *n>7* Teile keine praktische Bedeutung hat. Aus diesem Grund wurde die Ermittlung der GK für n>16 nicht weiter durchgeführt.
Es wird mit der Karatsuba-*n-TD* angefangen.

4.3.1. Ermittlung der Karatsuba-n-TD, mit *n≤256*

Der Produkt-Ausdruck der Karatsuba-Multiplikation für n-Bits Polynome bei ihrer Teilung in n Segmente ist für jedes *n* einzigartig und kann nicht als eine Funktion von den Segmenten, wie z.B. bei der klassischen (siehe **(98)**) oder generalisierten Karatsuba-MM (siehe **(105)**), ermittelt werden. Die Gatter-Komplexität der iterativ optimierten Berechnung der Karatsuba-n-TD kann trotzdem als eine Funktion von der Polynom-Länge *n* ermittelt werden.

Zuerst wird die Erstellung der Karatsuba-n-TD erklärt.
Bei der originalen Karatsuba-MM ist die Aufteilung der Operanden in nur zwei Segmente vorgesehen. Mittels der rekursiven Anwendung der Karatsuba-Multiplikations-Formel kann der Produkt-Ausdruck für 2^k-Bits Polynome ermittelt werden. Die Multiplikations-Formeln für *n*-Bits Polynome, $1<n<2^k$, können aus dem Produkt-Ausdruck für 2^k-Bits Polynome durch das Nullsetzen der höchstwertigen Bits der beiden Multiplikanden abgeleitet werden. Diese Ableitung wird an dem Beispiel der 4-Bits-Karatsuba-MF (siehe **(60)**) erklärt.

Formel **(60)** ist der algebraische Ausdruck der Karatsuba-Multiplikations-Formel der 4-Bits Polynome, der mit der rekursiv angewendeten Karatsuba-Multiplikations-Formel berechnet wurde. **Tabelle 13** ist die TD dieser Formel, d.h. die Karatsuba-4-TD.

Tabelle 13: Karatsuba-4-TD

$a_0 \cdot b_0 = p_1$				⊕	⊕	⊕	⊕
$a_1 \cdot b_1 = p_2$			⊕	⊕	⊕	⊕	
$a_2 \cdot b_2 = p_3$		⊕	⊕	⊕	⊕		
$a_3 \cdot b_3 = p_4$	⊕	⊕	⊕	⊕			
$(a_0 \oplus a_2) \cdot (b_0 \oplus b_2) = p_5$				⊕	⊕		
$(a_1 \oplus a_3) \cdot (b_1 \oplus b_3) = p_6$			⊕	⊕			
$(a_0 \oplus a_1) \cdot (b_0 \oplus b_1) = p_7$				⊕		⊕	
$(a_2 \oplus a_3) \cdot (b_2 \oplus b_3) = p_8$		⊕		⊕			
$(a_0 \oplus a_1 \oplus a_2 \oplus a_3) \cdot (b_0 \oplus b_1 \oplus b_2 \oplus b_3) = p_9$				⊕			
	c_6	c_5	c_4	c_3	c_2	c_1	c_0

Aus der 4-TD ist die 3-TD durch das Nullsetzen von $a_3=0$ und $b_3=0$ ableitbar. **Abbildung 14** stellt die Ableitung dar. Nachdem das höchstwertige Bit auf 0 gesetzt wurde, kommen manche TP in der linken 4-TD-Spalte doppelt vor. Aufgrund der Eigenschaft der XOR-Operation $a \oplus a=0$ werden bestimmte Zellen gelöscht. Der Inhalt solcher Zellen wird in **Abbildung 14** grün gekennzeichnet. Analog kann aus der 3-TD die 2-TD durch das Nullsetzen von $a_2=0$ und $b_2=0$ abgeleitet werden.

$a_0 \cdot b_0 = p_1$				⊕	⊕	⊕	⊕
$a_1 \cdot b_1 = p_2$			⊕	⊕	⊕	⊕	
$a_2 \cdot b_2 = p_3$		⊕	⊕	⊕	⊕		
$a_3 \cdot b_3 = p_4 = 0$	⊕	⊕	⊕	⊕			
$(a_0 \oplus a_2) \cdot (b_0 \oplus b_2) = p_5$				⊕	⊕		
$(a_1 \oplus 0) \cdot (b_1 \oplus 0) = p_6 = p_2$			⊕	⊕			
$(a_0 \oplus a_1) \cdot (b_0 \oplus b_1) = p_7$				⊕		⊕	
$(a_2 \oplus 0) \cdot (b_2 \oplus 0) = p_8 = p_3$		⊕		⊕			
$(a_0 \oplus a_1 \oplus a_2 \oplus 0) \cdot (b_0 \oplus b_1 \oplus b_2 \oplus 0) = p_9$				⊕			
	c_6	c_5	c_4	c_3	c_2	c_1	c_0

Schritt1: $a_3=0$, $b_3=0$ in 4-TD

$a_0 \cdot b_0 = p_1$				⊕	⊕	⊕	⊕
$a_1 \cdot b_1 = p_2$					⊕	⊕	
$a_2 \cdot b_2 = p_3$			⊕		⊕		
$p_4 = 0$							
$(a_0 \oplus a_2) \cdot (b_0 \oplus b_2) = p_5$				⊕	⊕		
$a_1 \cdot b_1 = p_2$							
$(a_0 \oplus a_1) \cdot (b_0 \oplus b_1) = p_7$				⊕		⊕	
$a_2 \cdot b_2 = p_3$							
$(a_0 \oplus a_1 \oplus a_2) \cdot (b_0 \oplus b_1 \oplus b_2) = p_9$				⊕			
	c_6	c_5	c_4	c_3	c_2	c_1	c_0

Schritt2: Das Löschen der doppelten Zellen

a) Ableitung der 3-TD aus der 4-TD

$a_0 \cdot b_0 = p_1$		⊕	⊕	⊕	⊕
$a_1 \cdot b_1 = p_2$			⊕	⊕	
$a_2 \cdot b_2 = p_3 = 0$	⊕		⊕		
$(a_0 \oplus 0) \cdot (b_0 \oplus 0) = p_5 = p_1$		⊕	⊕		
$(a_0 \oplus a_1) \cdot (b_0 \oplus b_1) = p_7$		⊕		⊕	
$(a_0 \oplus a_1 \oplus 0) \cdot (b_0 \oplus b_1 \oplus 0) = p_9 = p_7$		⊕			
	c_4	c_3	c_2	c_1	c_0

Schritt1: $a_2=0$, $b_2=0$ in 3-TD

$a_0 \cdot b_0 = p_1$				⊕	⊕
$a_1 \cdot b_1 = p_2$			⊕	⊕	
$p_3 = 0$					
$(a_0 \oplus 0) \cdot (b_0 \oplus 0) = p_5 = p_1$					
$(a_0 \oplus a_1) \cdot (b_0 \oplus b_1) = p_7$				⊕	
$(a_0 \oplus a_1 \oplus 0) \cdot (b_0 \oplus b_1 \oplus 0) = p_9 = p_7$					
	c_4	c_3	c_2	c_1	c_0

Schritt2: Das Löschen der doppelten Zellen

b) Ableitung der 2-TD aus der 3-TD

$a_0 \cdot b_0 = p_1$		⊕	⊕
$a_1 \cdot b_1 = p_2 = 0$	⊕	⊕	
$(a_0 \oplus 0) \cdot (b_0 \oplus 0) = p_7 = p_1$		⊕	
	c_2	c_1	c_0

Schritt1: $a_1=0$, $b_1=0$ in 2-TD

$a_0 \cdot b_0 = p_1$			⊕
$p_2 = 0$			
$(a_0 \oplus 0) \cdot (b_0 \oplus 0) = p_7 = p_1$			
	c_2	c_1	c_0

Schritt2: Das Löschen der doppelten Zellen

c) Ableitung der 1-TD aus der 2-TD

Abbildung 14: Ableitung der 3-, 2- und 1-TD aus der 4-TD

Nach diesem Verfahren wurde die Ermittlung aller *n*-TD für *n*<256 aus der 256-TD, $256=2^8$, programmiert.

4.3.2. Besonderheit der Karatsuba-n-TD

Obwohl die TD der Karatsuba-MF für jedes *n* einzigartig ist, haben alle n-TD[6] eine gemeinsame Besonderheit, die den iterativ optimierten Ablaufplan der Produkt-Berechnung und seine GK bestimmen hilft. Um diese Besonderheit zu beschreiben, wurden die folgenden Bezeichnungen eingeführt:

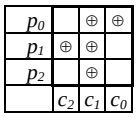
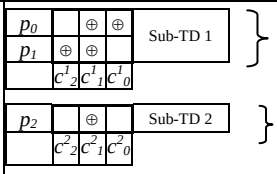
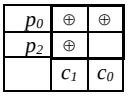
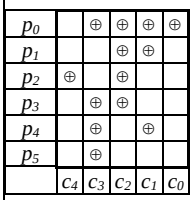
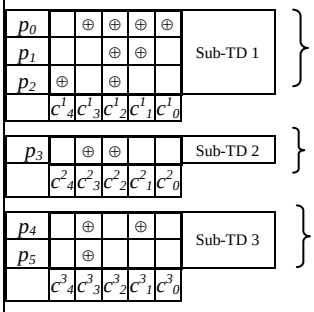
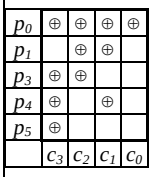
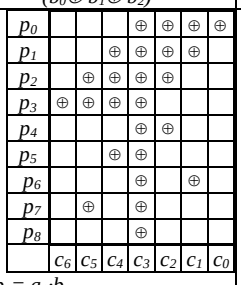
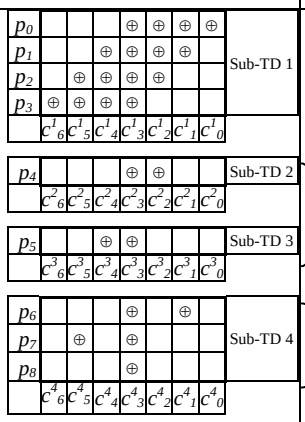
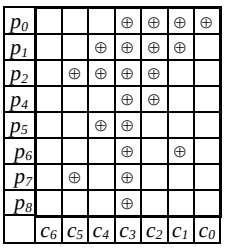
Als *Große Raute* einer n-TD (gekennzeichnet mit *n-GR*) wird eine Sub-TD, die aus *n* Zeilen mit TP $p_i=a_ib_i$, $0\le i\le n-1$ besteht, bezeichnet.
Als *nicht volle Große Raute* einer n-TD (gekennzeichnet mit *NV-n-GR*) wird eine Sub-TD, die aus *n-1* Zeilen mit TP $p_i=a_ib_i$, $0\le i\le n-2$ besteht, bezeichnet. Die *NV-n-GR* entspricht der *n-GR* ohne die Zeile mit dem TP p_{n-1}.
Als *nicht volle n*-TD (gekennzeichnet mit *NV-n-TD*) wird eine *n*-TD ohne die Zeile mit dem TP p_{n-1} bezeichnet.

Die Besonderheit der *Karatsuba-n-TD* ist:
Jede *n*-TD besteht aus der n-GR und weiteren Sub-TD, die zur *i*-TD oder zur *NV-i*-TD, $i<n$, identisch sind. Alle Zeilen einer Sub-TD (außer der *n*-GR) haben die gleiche Anzahl ununterbrochen gefüllter Zellen *L*.

Tabelle 14 zeigt die Aufteilung der n-TD für $1\le n\le 4$ in Sub-TD auf Basis dieser Besonderheit. Auf jede *i*-TD und *NV-i*-TD können die Kriterien ***(K)*** angewendet werden, weil die *n*-GR den *iterativen* TD-Teil bildet.

[6] In diesem Abschnitt bezieht sich diese Bezeichnung immer auf die Karatsuba-*n*-TD

Tabelle 14: Aufteilung aller n-TD, für *1≤n≤4,* in Sub-TD

	Karatsuba-n-TD	Teilung in Sub-TD	Inhalt der n-TD	*NV-n-TD*
1-TD	 $p_0 = a_0 \cdot b_0$	---	1-GR	---
2-TD	 $p_0 = a_0 \cdot b_0$ $p_1 = a_1 \cdot b_1$ $p_2 = (a_0 \oplus a_1) \cdot (b_0 \oplus b_1)$		2-GR eine Sub-TD, die zur 1-TD *äquivalent*[7] ist	 NV-2-TD: die Zeile mit TP p_1 fehlt.
3-TD	 $p_0 = a_0 \cdot b_0$ $p_1 = a_1 \cdot b_1$ $p_2 = a_2 \cdot b_2$ $p_3 = (a_0 \oplus a_2) \cdot (b_0 \oplus b_2)$ $p_4 = (a_0 \oplus a_1) \cdot (b_0 \oplus b_1)$ $p_5 = (a_0 \oplus a_1 \oplus a_2) \cdot$ $\cdot (b_0 \oplus b_1 \oplus b_2)$		3-GR eine Sub-TD, die zur 1-TD *äquivalent* ist eine Sub-TD, die zur 2^{NV}-TD *äquivalent* ist	 NV-3-TD: die Zeile mit TP p_2 fehlt.
4-TD	 $p_0 = a_0 \cdot b_0$ $p_1 = a_1 \cdot b_1$ $p_2 = a_2 \cdot b_2$ $p_3 = a_2 \cdot b_3$ $p_4 = (a_0 \oplus a_2) \cdot (b_0 \oplus b_2)$ $p_5 = (a_1 \oplus a_3) \cdot (b_1 \oplus b_3)$ $p_6 = (a_0 \oplus a_1) \cdot (b_0 \oplus b_1)$ $p_7 = (a_2 \oplus a_3) \cdot (b_2 \oplus b_3)$ $p_8 = (a_0 \oplus a_1 \oplus a_2 \oplus a_3) \cdot$ $\cdot (b_0 \oplus b_1 \oplus b_2 \oplus b_3)$		4-GR zwei Sub-TD, die zur 1-TD *äquivalent* sind eine Sub-TD, die zur 2-TD *äquivalent* ist	 NV-4-TD: die Zeile mit TP p_3 fehlt.

[7] Zwei Sub-TD werden als äquivalent bezeichnet, wenn durch Umformen einer Sub-TD die andere Sub-TD erhalten werden kann. Zum Umformen gehört das Zufügen (oder das Löschen) der leeren oder gleichgefüllten Spalten und das Umbenennen der Spalten.

Analog zum in **Tabelle 14** illustrierten Vorgehen wurde die Aufteilung der n-TDs, für $n \leq 32$, in ihre Sub-TDs durchgeführt. **Tabelle 15** zeigt, in welche Sub-TDs jede Karatsuba-*n*-TD aufgeteilt werden kann.

Tabelle 15: Aufteilung der Karatsuba-*n*-TD, für $n \leq 32$, in ihre Sub-TDs

n	**Sub-TD**																																
	n-GR	**1**	*NV-2*	**2**	*NV-3*	**3**	*NV-4*	**4**	*NV-5*	**5**	*NV-6*	**6**	*NV-7*	**7**	*NV-8*	**8**	*NV-9*	**9**	*NV-10*	**10**	*NV-11*	**11**	*NV-12*	**12**	*NV-13*	**13**	*NV-14*	**14**	*NV-15*	**15**	*NV-16*	**16**
1	1																																
2	1	1																															
3	1	1	1																														
4	1	2		1																													
5	1	2	1		1																												
6	1	2	2			1																											
7	1	3	1	1			1																										
8	1	4		2				1																									
9	1	4	1	1	1				1																								
10	1	4	2		2					1																							
11	1	4	3		1	1					1																						
12	1	4	4			2						1																					
13	1	5	3	1		1	1						1																				
14	1	6	2	2			2							1																			
15	1	7	1	3			1	1							1																		
16	1	8		4				2								1																	
17	1	8	1	3	1			1	1								1																
18	1	8	2	2	2				2									1															
19	1	8	3	1	3				1	1									1														
20	1	8	4		4					2										1													
21	1	8	5		3	1				1	1										1												
22	1	8	6		2	2					2											1											
23	1	8	7		1	3					1	1											1										
24	1	8	8			4						2												1									
25	1	9	7	1		3	1					1	1												1								
26	1	10	6	2		2	2						2													1							
27	1	11	5	3		1	3						1	1													1						
28	1	12	4	4			4							2														1					
29	1	13	3	5			3	1						1	1														1				
30	1	14	2	6			2	2							2															1			
31	1	15	1	7			1	3							1	1															1		
32	1	16		8				4								2																1	

Die Zahlen in **Tabelle 15** weisen folgende Regelmäßigkeit auf: sie bilden „Strahlen“. Die Strahlen werden von rechts mit ‚0’ beginnend durchnummeriert um sie bezeichnen zu können. Der rechte Strahl besteht nur aus Einsen ($1=2^0$) und alle durch ihn repräsentierten Sub-TD haben $L=2^0=1$ ununterbrochen gefüllten Zellen je Zeile. Aus diesem Grund werden alle seine Sub-TD dem *separaten* TD-Teil (siehe Kriterien ***(K)***) zugeordnet.

Der nächste Strahl wird von den gleichen Zahlen-Gruppen gebildet. Jede Gruppe besteht aus 3 Zahlen, die vertikal angeordnet sind: 1, 2, 1. Die Anzahl der ununterbrochen gefüllten Zellen der entsprechenden Sub-TD ist $L=2^1$.

Jeder weitere Strahl, der Strahl Nummer *i*, wird auch von Zahlen-Gruppen gebildet. Die Gruppe beginnt mit der Zahl *1*, anschließend erhöht sich der Wert pro Zeile um 1 bis zum Wert 2^i, anschließend verringert sich dieser Wert wieder in Einer-Schritten je Zeile bis zum Wert 1. Alle Sub-TD der Strahlen mit Nummern mit $i>0$ gehören zum *iterativen* Teil.

Die **Tabelle 15** kann für beliebig große *n* ohne Kenntnisse der *n*-TD erweitert werden. Der Erweiterungsprozess wurde in C programmiert. Die Korrektheit der Ausfüllung der **Tabelle 15** wurde auf *n*-TD, $32<n\leq 256$, geprüft.

4.3.3. Berechnungsalgorithmus

Tabelle 15 zeigt, aus welchen und auch aus wie vielen *i-TD* und *NV-i*-TD eine *n*-TD besteht, für $i<n$. Aus der Zuordnung der *i-TD* (oder *NV-i*-TD) zu einem der *i* Strahlen ergibt sich, nach welcher Art der Berechnung – *iterativ* oder *separat* – die *i*-TD berechnet werden sollte. Dieses Wissen ermöglicht es, die Gatter-Komplexität der *n*-TD für beliebig große *n* folgendermaßen zu ermitteln:

$$GC^{Karatsuba}_{n-TD} = \#AND[n-TD]_{AND} + \#XOR[n-TD]_{XOR},$$

mit

$$\#AND[n-TD] = \#AND[n-GR] + \sum_{i=1}^{n-1}\left(\begin{array}{l}\#AND[i-TD]\cdot Anzahl\_i-TD + \\ +\#AND[NV-i-TD]\cdot Anzahl\_NV-i-TD\end{array}\right)$$

$$\#XOR[n-TD] = \#XOR_{\substack{\text{Berechnung}\\ \text{der Operanden}\\ \text{für TM}}}[n-TD] + \#XOR_{\text{Spalten}}[n-TD]$$

$$\#XOR_{\text{Spalten}}[n-TD] = \#XOR[n-GR] + \sum_{i=1}^{n-1}\left(\begin{array}{l}\left(\begin{array}{l}\#XOR_{\text{Spalten}}[i-TD] + \\ +\begin{cases}2i-1, & i-TD\in separatem\ n\text{-TD-Teil}\\ 2\cdot(2i-1), & i-TD\in itertivem\ n\text{-TD-Teil}\end{cases}\end{array}\right)\cdot Anzahl\_i-TD + \\ +\sum_{k=1}^{Anzahl\_i-TD} F_k[i-TD]\end{array}\right) +$$

$$+\sum_{i=1}^{n-1}\left(\begin{array}{l}\left(\begin{array}{l}\#XOR_{\text{Spalten}}[NV-i-TD] + \\ +\begin{cases}2i-2, & NV-i-TD\in sep.\ n\text{-TD-Teil}\\ 2\cdot(2i-2), & i-TD\in iter.\ n\text{-TD-Teil}\end{cases}\end{array}\right)\cdot Anzahl\_NV-i-TD + \\ +\sum_{k=1}^{Anzahl\_NV-i-TD} F_k[NV-i-TD]\end{array}\right) \quad \textbf{(112)}$$

Die Anzahl der *i*-TD und der *NV-i*-TD, aus denen die *n*-TD besteht, $i<n$, werden in Formel **(112)** als *Anzahl_i-TD* und *Anzahl_NV-i-TD* bezeichnet. Die Anzahl der AND-Gatter für die Berechnung aller TP, aus denen die GR besteht, ist gleich *n*. Formel **(115)** beschreibt die Anzahl der XOR-Gatter für die Ermittlung der

Operanden für die restlichen Teil-Multiplikationen unter der Bedingung, dass jeder Operand außer a_i und b_i, $0 \le i < n$, mit nur einem XOR-Gatter (d.h. mit minimalem XOR-Aufwand) aus bereits berechneten Operanden-Werten ermittelt werden kann[8].

$$\#XOR_{\substack{\text{Berechnung}\\ \text{der Operanden}\\ \text{für TM}}}\left[n-TD\right] = \left(\#AND\left[n-TD\right]-n\right)\cdot 2 \qquad \textbf{(113)}$$

Der XOR-Aufwand der Berechnung der Spalten der *n*-TD besteht aus den XOR-Aufwänden der Berechnung aller beinhalteten Sub-TDs und der Addition (XOR) der ermittelten Sub-TD-Spalten-Werte. In dieser Arbeit wurde festgestellt (siehe Kapitel 4.3.4), dass die optimale Berechnung der *n*-TD immer mit der *iterativen* Berechnung eines *n*-GR-Teils beginnen wird. Für jede weitere *i*- oder *NV-i*-Sub-TD kann nach den Kriterien ***(K)*** entweder eine *iterative* oder *separate* Berechnung festgelegt werden. Die *i*-Sub-TD hat (*2i-1*) und die *NV-i*-Sub-TD hat (*2i-2*) Spalten. Falls diese Sub-TD zum *separaten n*-TD-Teil gehört, ist nur 1 XOR-Gatter pro Sub-TD-Spalte für die optimale Addition (XOR) dieses, bereits ermittelten, Spalten-Wertes nötig. 2 XOR-Gatter pro Sub-TD-Spalte sind nötig, wenn diese Sub-TD zum *iterativen n*-TD-Teil gehört. Die Variable F_k in **(112)** berücksichtigt den Sonderfall, in dem eine Spalte der Sub-TD vom *iterativen* TD-Teil mit nur 1 XOR-Gatter addiert (geXORt) werden kann (siehe Sonderfälle der Kriterien ***(K)***). Falls alle *i*-TD-Spalten (*NV-i-TD*-Spalten) 2 XOR-Gatter benötigen, ist $F_k=0$, sonst $F_k=-1$. Damit wird jede Spalte jeder beinhalteten *i*- und *NV-i*-Sub-TD mit minimalem XOR-Aufwand in **(112)** addiert (geXORt). Wenn alle Sub-TD auch mit minimalem XOR-Aufwand, d.h. *optimal*, berechnet wurden, wird eine solche Berechnung der *n*-TD als *iterativ optimierte* Berechnung bezeichnet.
Wenn der XOR-Aufwand #XOR[*n-GR*] der Berechnung der *n*-GR nach einem optimalen Ablaufplan für beliebige *n* algebraisch oder algorithmisch ermittelt werden kann, kann auch der minimale XOR-Aufwand der Berechnung der *n*-TD-Spalten $\#XOR_{Spalten}$[*n-GR*] und (nachfolgend) die GC der iterativ optimierten *n*-TD-Berechnung nach Formel **(112)** für beliebig große *n* algorithmisch ermittelt werden. ***Algorithmus 3*** stellt die Ermittlung des $\#XOR_{Spalten}$[*n-GR*] dar.

Die Ermittlung des XOR-Aufwandes der Berechnung der *n*-TD-Spalten beginnt mit der Initialisierung des XOR-Aufwandes für die Berechnung der 1-TD- und 2-TD. Anschließend erfolgt die Ermittlung konsequent bis zu der gewünschten Polynom-Länge *n*. Die iterativ optimierte Berechnung der *n*-GR und die Herleitung ihrer XOR-Aufwandes #XOR[*n-GR*] wird als eine selbstständige Aufgabe im nächsten Kapitel gelöst.

[8] Die Erfüllung dieser Bedingung wurde in dieser Arbeit nicht bewiesen, aber für alle *n*-TD, $n \le 256$, geprüft.

Algorithmus 3

Input: Polynom-Länge *n,* **Tabelle 15,** GC nach **(112)**
Output: $\#XOR_{\text{Spalten}}[i-TD]$, *0≤i≤n*

Initialisation
#XOR[*1-TD*]=0
#XOR[*2-TD*]=2,
#XOR[*NV-2-TD*]=1

Berechnung
for *each i| 3≤i≤n* ***//*** TD aller kleineren Polynom-Längen werden berechnet
- **ermitteln** *#XOR[i-GR]; #XOR[NV-i-GR];*
- *#XOR[i-TD] = #XOR[i-GR]*
for *each j|1≤j≤i*
if (*Anzahl_j-TD≠0*)
if (*j-TD zum separaten i-TD-Teil gehört*)***//***rechter „Strahl“ aus Tabelle 15
#XOR[i-TD]+= Anzahl_j-TD · (*#XOR[j-TD]* +(2*j*-1))
else
#XOR[i-TD]+= Anzahl_j-TD · (*#XOR[j-TD]* +2·(2*j*-1))
end if
end if
if (*Anzahl_NV-j-TD≠0*)
if (*NV-j-TD separatem i-TD-Teil gehört*)***//***rechter „Strahl“ aus Tabelle 15
#XOR[i-TD] += *Anzahl_NV-j-TD* · (*#XOR[NV-j-TD]* +(2*j*-2))
else
#XOR[i-TD] += *Anzahl_NV-j-TD* · (*#XOR[NV-j-TD]* +2·(2*j*-2))
end if
end if
end for

$korrektur = -(Anzahl\_der\_Strahlen\_in\_Zeile\_i - 1)$***//***Sonderfälle der Kriterien **(*K*)**
#XOR[i-TD]+= korrektur
#XOR[NV-i-TD]=#XOR[i-TD] - *#XOR[i-GR]* + *#XOR[NV-i-GR]*

end for

4.3.4. Berechnung der Großen Raute der n-TD

Da die ganze *n*-TD als ein Set der TD kleinerer Polynom-Längen darstellbar ist, wurde die Darstellbarkeit der *n*-GR mittels der GR der kleineren Polynom-Längen geprüft. Die GR aller *n*-TD wurden bis zur Länge von 32-Bits Polynome analysiert, um die gemeinsamen Merkmale zu finden. Die Ergebnisse der Analyse wurden an weiteren *n*-TD, für *32<n≤256,* überprüft.

Es wurden folgende Merkmale festgestellt:
Merkmal 1:
Die rechte n-GR-Seite, zu der die n-GR-Spalten $c^{n\text{-}GR}{}_0 \ldots c^{n\text{-}GR}{}_{n-1}$ gehören, kann bei allen *n*-GR mit Hilfe der gleichen Regeln beschrieben werden: die n-GR-Spalte $c^{n\text{-}GR}{}_i$ hat insgesamt (*i+1*) gefüllte Zellen mit TP p_j (*0≤j≤i*) (siehe **(114)**).

$$c_i^{n-GR} = \bigoplus_{j=0}^{i} p_j, \quad \left|P_{c_i^{n-GR}}\right| = i+1 \tag{114}$$

Merkmal 2:
Die linke n-GR-Seite, zu der die GR-Spalten $c^{n\text{-}GR}{}_{2n-2} \ldots c^{n\text{-}GR}{}_n$ gehören, kann mittels der GR der kleineren Polynom-Längen und der anderen, von den gefüllten Zellen gebildeten, geometrischen Figuren dargestellt werden. Die Anzahl der Figuren und deren geometrische Formen sind von der Binärdarstellung der Polynom-Länge *n* abhängig.

Merkmal 1 und *2* belegen, dass die Berechnungen der linken und der rechten Seiten aller n-GR einzeln durchgeführt und optimiert werden dürfen.

4.3.4.1. Rechte Seite Großer Raute

Aus *Merkmal 1* (siehe **(114)**) folgt:

$$P_{c_0^{n-GR}} \subset P_{c_1^{n-GR}} \subset P_{c_2^{n-GR}} \subset \ldots \subset P_{c_{n-1}^{n-GR}}, \text{ wobei } \left|P_{\Delta(c_i, c_{i-1})}\right| = 1,\ 0 < i \le n-1 \tag{115}$$

Aus **(115)** aufgrund **(90)-(95)** folgt, dass der Ablaufplan mit dem minimalen XOR-Aufwand der *voll iterative* Ablaufplan **(116)** ist:

$$\overset{separat}{c_0^{n-GR}} \rightarrow c_1^{n-GR} \rightarrow \ldots \rightarrow c_{n-1}^{n-GR} \tag{116}$$

Der XOR- Aufwand der Berechnung nach Ablaufplan **(116)** ist folgender:

$$\#XOR_{n-GR}^{\overset{optimal}{rechte\ Seite}} = \#XOR_{c_0^{n-GR} \ldots c_{n-1}^{n-GR}}^{\overset{optimal}{}} = \#XOR_{c_0^{n-GR} \ldots c_{n-1}^{n-GR}}^{\overset{voll\ iterativ}{}} =$$
$$= \#XOR_{c_0^{n-GR}}^{\overset{separat}{}} + \sum_{i=1}^{n-1} \#XOR_{c_i^{n-GR}}^{\overset{iterativ}{}} = \left(\left|P_{c_0^{n-GR}}\right| - 1\right) + \sum_{i=1}^{n-1} \left|P_{\Delta\left(c_i^{n-GR}, c_{i-1}^{n-GR}\right)}\right| = 0 + \sum_{i=1}^{n-1} 1 = n-1 \tag{117}$$

Abbildung 15 zeigt die GR der TD des PP der 7-, 8-, 10- und 13-Bits großen Polynome, um das *Merkmal* 1 (siehe **(114)**) und den optimalen Ablaufplan **(116)** zu illustrieren.

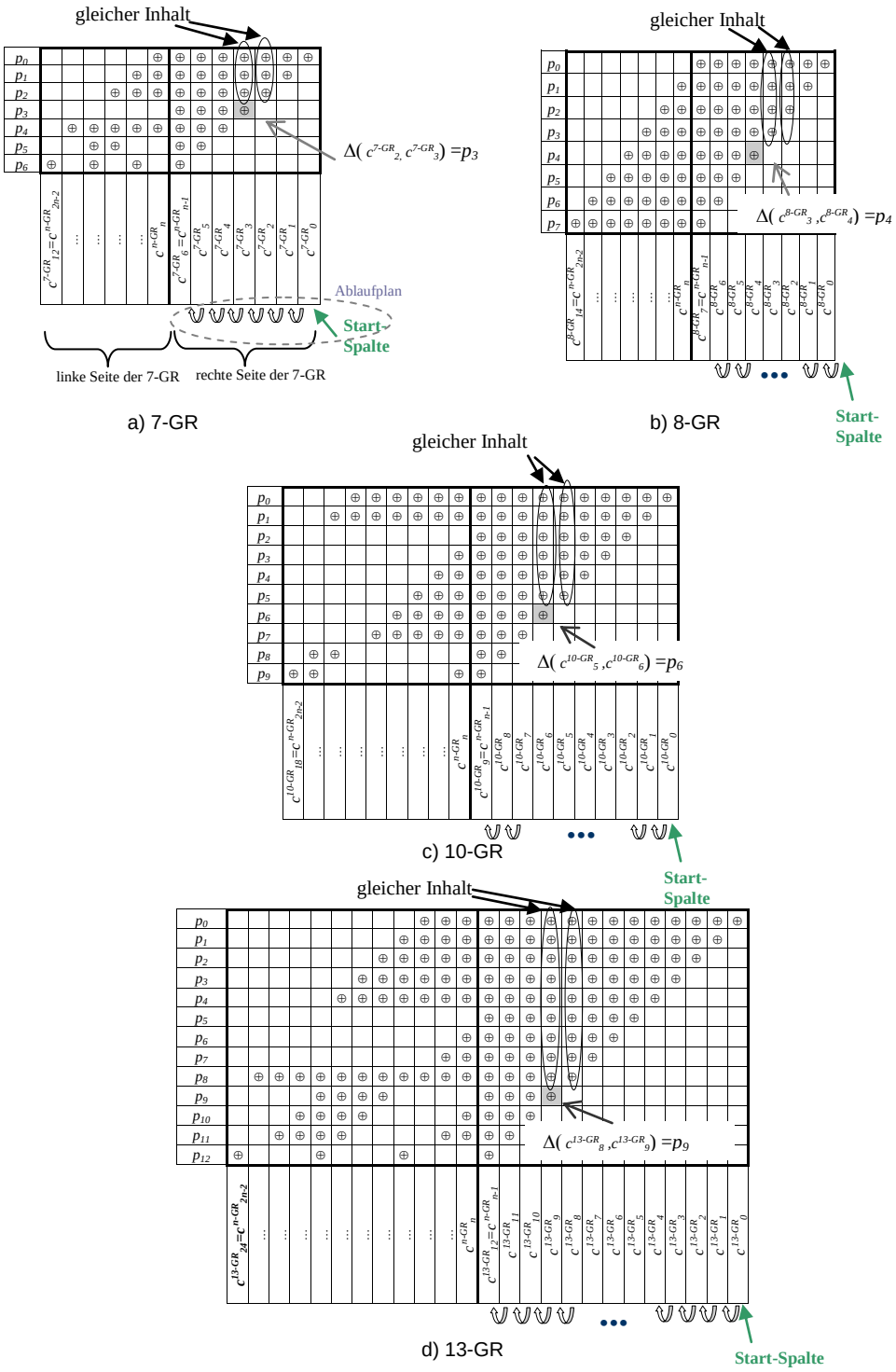


Abbildung 15: Karatsuba-n-GR für 7-, 8-, 10- und 13-Bits-Polynome.
Jede Spalte der rechten Seite aller n-GR beinhaltet alle TP ihrer rechten Nachbar-Spalte und ein weiteres Teil-Produkt

4.3.4.2. Linke Seite Großer Raute

Es wurde festgestellt (siehe *Merkmal 2*), dass alle *n*-GR in 4 Gruppen aufgeteilt werden können. Diese Tatsache erleichtert die Ermittlung der GK der *n*-GR als eine Funktion von *n*: anstatt der Suche des optimalen Ablaufplanes für jede einzelne Polynom-Länge *1≤n≤256* braucht nur ein optimaler Ablaufplan und seine GK pro Gruppe ermittelt zu werden. Jetzt werden die Eingruppierungskriterien erklärt und der optimale Ablaufplan der Berechnung der linken GR-Seite jeder Gruppe mit den jeweiligen XOR-Aufwänden ermittelt. Bei der Suche des optimalen Ablaufplanes werden die Merkmale jeder n-GR-Gruppe am Beispiel eines typischen Vertreters illustriert.

Es wurde folgendes festgestellt:
Die gefüllten Zellen bilden in allen GR geometrische Figuren. Es gibt einen Zusammenhang zwischen der Binärdarstellung der Polynom-Länge *n* und der Anzahl dieser Figuren und deren geometrischen Formen. Jetzt wird *n* ($2^q \le n < 2^{q+1}$) als Binarzahl dargestellt, um diesen Zusammenhang und die Eingruppierung der *n*-GR zu erklären. Formel **(118)** ist die Binärdarstellung der *n*:

$$n = \underbrace{\delta_q}_{=1}\delta_{q-1}\ldots\delta_1\delta_0, \quad \delta_i \in \{0,1\}$$

(118)

Angenommen, alle höchstwertigen *(j+1)* Bits sind Eins und die nächste Stelle (mit Index *q-(j+1)*) ist mit Null besetzt. Dann ist *n* auch wie folgt darstellbar:

$$n = \underbrace{11\ldots1}_{j+1\,\text{Bits}}\underbrace{0}_{1\,\text{Bit}}\underbrace{\delta_{q-(j+2)}\ldots\delta_0}_{rest} = 2^q + \ldots + 2^{q-j} + r = \sum_{i=0}^{j} 2^{q-i} + r, \quad 0 \le r < 2^{q-j-1}$$

(119)

Das Aussehen der *n*-GR ist von den Zahlen *j* und *r* abhängig. Alle n-GR können in 4 Gruppen aufgeteilt werden. **Tabelle 16** bietet eine kurze Beschreibung der n-GR-Gruppen.
Jedes *n* kann eindeutig einer der n-GR-Gruppen zugeordnet werden. Die optimalen Ablaufpläne und deren XOR-Aufwände für *n≤3* sind eindeutig. Aus diesem Grund müssen nur die Fälle für *n>3* untersucht werden.
Jetzt wird jede n-GR-Gruppe einzeln analysiert.

Tabelle 16: Karatsuba-n-GR-Gruppen

n-GR-Gruppe	*j*	*r*	*n*	Beschreibung	Beispiel *n*
1	0	0	$n=2^q+0$	Nur die höchste Stelle ist mit Eins besetzt und alle anderen mit Nullen	$n=1_2=1$, $n=10_2=2$, $n=100_2=4$, $n=1000_2=8$,…$n=100000_2=32$
2	0	>0	$n=2^q+r$	Die höchste Stelle ist mit Eins besetzt, die nächste Stelle mit Null, und alle anderen Stellen bilden *r*	$n=101_2=5$, $n=1001_2=9$, $n=1010_2=10$, … $n=10001_2=17$, $n=10010_2=18$, $10011_2=19$,…
3	>0	0	$n=2^q+2^{q-1}+\ldots+2^{q-j}+0$	Die höchsten *(j+1)* Stellen sind mit Einsen und alle anderen mit Nullen besetzt	$n=11_2=3$, $n=110_2=6$, $n=1100_2=12$,…, $n=11100_2=28$, …
4	>0	>0	$n=2^q+2^{q-1}+\ldots+2^{q-j}+r$	Die höchsten *(j+1)* Stellen sind mit Einsen besetzt, dienächste Stelle mit Null, und alle anderen Stellen bilden *r*	$n=1101_2=13$, $n=11001_2=25$, $n=11010_2=26$, $n=11011_2=27$, …

n-GR-Gruppe 1

Zu dieser Gruppe gehören alle Polynome, deren Länge ist $n=2^q+0$. Jede *n*-GR-Zeile hat *n* ununterbrochen gefüllte Zellen, die zusammen eine geometrische Figur bilden, die einem Parallelogramm ähnlich ist. Die Länge seiner Grundseite ist die Anzahl der ununterbrochen gefüllten Zellen *L=n*. Die dazugehörige Höhe besteht auch aus *n* gefüllten Zellen: *h=n*. **Abbildung 15-b)** stellt die GR eines typischen Vertreters der Gruppe 1, 8-GR, dar.

Die linke GR-Seite, zu der die GR-Spalten $c^{n\text{-}GR}{}_{2n-2}\ldots c^{n\text{-}GR}{}_{n}$ gehören, kann bei allen *n*-GR dieser Gruppe mit den gleichen Regeln beschrieben werden: die GR-Spalte $c^{n\text{-}GR}{}_{i}$, $2n-2\geq i\geq n$, hat insgesamt $2n-(i+1)$ gefüllte Zellen mit TP p_j, $i-(n-1)\leq j\leq n-1$.

$$c_i^{n-GR} = \bigoplus_{j=i-(n-1)}^{n-1} p_j$$

(120)

D.h., jede GR-Spalte $c^{n\text{-}GR}{}_{i}$ ($2n-3\geq i\geq n$) beinhaltet in sich alle TP ihrer linken Nachbar-Spalte $c^{n\text{-}GR}{}_{i+1}$ und noch ein Teil-Produkt. Formel **(121)** beschreibt diese Tatsache.

$$P_{c_{2n-2}^{n-GR}} \subset P_{c_{2n-3}^{n-GR}} \subset P_{c_{2n-4}^{n-GR}} \subset \ldots \subset P_{c_n^{n-GR}}, \text{ mit } \left| P_{\Delta\left(c^{n-GR}{}_i, c^{n-GR}{}_{i+1}\right)} \right| = 1$$

(121)

Aus **(121)** aufgrund **(80)-(84)** folgt, dass der Ablaufplan mit dem minimalen XOR-Aufwand der *voll iterative* Ablaufplan **(122)** ist:

$$c_{2n-2}^{n-GR} \overset{separat}{\rightarrow} c_{2n-3}^{n-GR} \rightarrow \ldots \rightarrow c_{n+1}^{n-GR} \rightarrow c_n^{n-GR}$$

(122)

Abbildung 16 stellt die linke Seite der 8-GR dar und illustriert den Ablaufplan **(122)**.

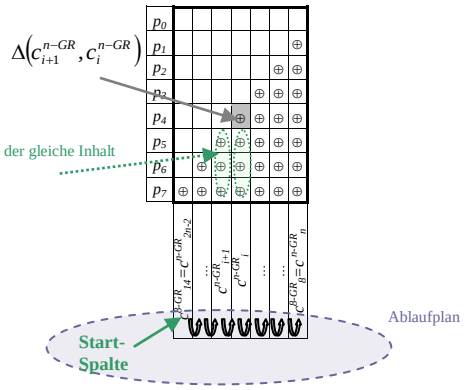


Abbildung 16: linke Seite der 8-GR, $n=8=2^q+0=1000_2 \Rightarrow q=3, r=0$

Der XOR- Aufwand des Ablaufplanes **(122)** ist folgender:

$$\#\,XOR_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}}^{optimal} = \#\,XOR_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}}^{voll\ iterativ} = \#\,XOR_{c_{2n-2}^{n-GR}}^{separat} + \sum_{i=n}^{2n-3} \#\,XOR_{c_i^{n-GR}}^{iterativ} =$$

$$= 0 + \sum_{i=n}^{2n-3} \left| P_{\Delta\left(c_i^{n-GR}, c_{i+1}^{n-GR}\right)} \right| = \sum_{i=n}^{2n-3} 1 = n-2$$

(123)

Bei *nicht vollen* GR bleibt der Ablaufplan **(122)** optimal. Der XOR-Aufwand ist wie folgt:

$$\#\,XOR_{c_{2n-3}^{NV-n-GR}\ldots c_n^{NV-n-GR}}^{\substack{optimal \\ nicht\ voll\ GR}} = \#\,XOR_{c_{2n-3}^{NV-n-GR}\ldots c_n^{NV-n-GR}}^{voll\ iterativ} =$$

$$= \#\,XOR_{c_{2n-3}^{NV-n-GR}}^{separat} + \sum_{i=n}^{2n-4} \#\,XOR_{c_i^{n-GR}}^{iterativ} = 0 + \sum_{i=n}^{2n-4} \left| P_{\Delta\left(c_i^{n-GR}, c_{i+1}^{n-GR}\right)} \right| = \sum_{i=n}^{2n-4} 1 = n-3$$

(124)

n-GR-Gruppe 2

Zu dieser Gruppe gehören alle Polynome, deren Länge ist $n=2^q+r,\ r>0$. Die **Abbildung 17** stellt die n-GR eines typischen Vertreters der n-GR-Gruppe 2 dar, die 19-GR.

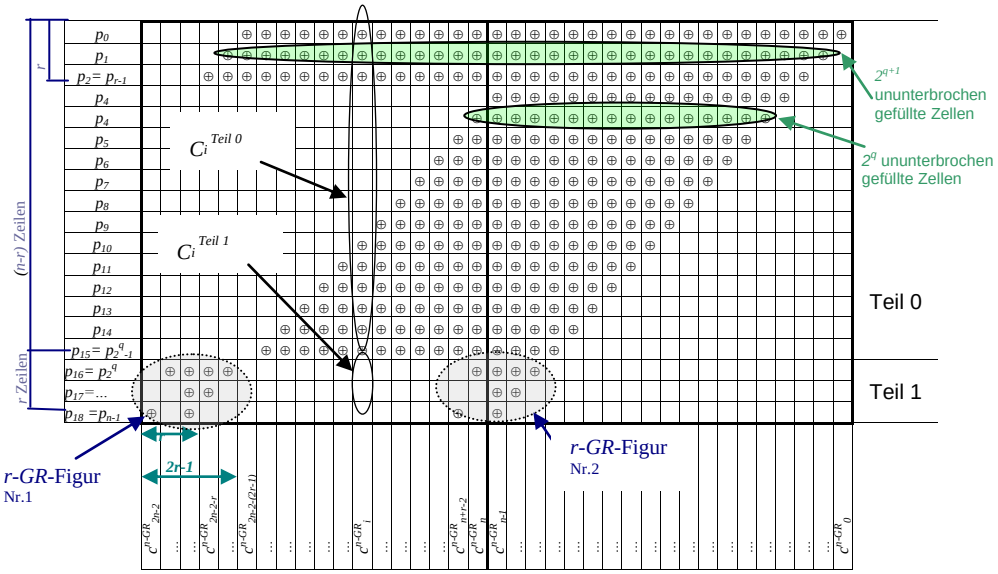


Abbildung 17: Vertreter der n-GR-Gruppe 2, 19-GR, $n=19=2^4+3 \Rightarrow q=4, r=3$

n-GRs dieser Gruppe bestehen aus 2 Teilen:

• **Teil 0** besteht aus p_0 … $p_2{}^q{}_{-1}$- Zeilen, wobei jede Zeile p_0 … p_{r-1} die $L=2^{q+1}$ ununterbrochen gefüllte Zellen hat und jede Zeile p_r … $p_2{}^q{}_{-1}$ die $L=2^q$. Die Teil-Spalten, welche zum Teil 0 gehören, werden mit $c_i^{n\text{-}GR\text{-}Teil0}$ bezeichnet (siehe **Abbildung 17**).

• **Teil 1** besteht aus $p_2{}^q$ … p_{n-1}- Zeilen, die die zwei *r-GR*-Figuren, *r-GR*-Figur Nr.1 und die *r-GR*-Figur Nr.2 beinhalten (siehe **Abbildung 17**). Die *r-GR*-Figur Nr.1 befindet sich in den Spalten $c^{n\text{-}GR}{}_{2n-2}$, … ,$c^{n\text{-}GR}{}_{2n-2-(2r-1)}$. Die *r-GR*-Figur Nr.2 befindet sich in den Spalten $c^{n\text{-}GR}{}_{n-1-r+1}$, …, $c^{n\text{-}GR}{}_{n-1+(r-1)}$, wobei sich ihre mittlere Spalte in der mittleren Spalte der ganzen *n*-GR befindet.

Für die linken *n-1* Spalten[9] des Teils 0 gilt folgendes: linke *r* Spalten sind leer, und jede weitere nach rechts folgende Spalte beinhaltet in sich alle TP ihrer linken Nachbar-Spalte und ein weiteres Teil-Produkt. Analog zu **(121)** ist hier die *iterative* (evtl. *voll iterative*) Berechnung optimal.

Die Frage, was der optimale Ablaufplan der Berechnung des Teils 1 ist, kann nicht eindeutig beantwortet werden. Einerseits kann die linke Seite der *r-GR*-Figur Nr.1[10] als eine selbstständige TD berechnet werden, weil die entsprechenden Spalten

[9] d.h. für die Spalten des Teils 0, die zur linken Seite der n-GR gehören

[10] d.h. die *r-1* linken Spalten der r-GR-Figur Nr.1

aus dem Teil 0 leer sind[11]. Diese Tatsache begünstigt die *separate* Berechnung des Teils 1, weil in diesem Fall der XOR-Aufwand der optimalen Berechnung der linken *r* Spalten der *n*-GR gleich $\#XOR^{optimal}_{\substack{linke\ Seite \\ r-GR-Figur}}$ ist. Für die *iterative* Berechnung spricht aber das Merkmal 1 (siehe **(114)**), dass für die rechte Seite der *r-GR*-Figur Nr.1 gilt. Im Fall der *iterativen* Berechnung der ganzen n-GR werden die entsprechenden Spalten der *r-GR*-Figur Nr.1 nach dem Ablaufplan berechnet:

$$\underbrace{c^{r-GR}_{r-1}}_{Start-Spalte} \rightarrow c^{r-GR}_{r-2} \rightarrow \ldots \rightarrow c^{r-GR}_{0} \qquad \textbf{(125)}$$

Die Möglichkeit, den Teil 1 nur teilweise *iterativ* zu berechnen, muss auch untersucht werden. Insgesamt gibt es die folgenden 8 Kombinationen:

r-GR-Figur Nr.1		*r-GR*-Figur Nr.2
linke Seite, d.h. linke *(r-1)* Spalten	rechte Seite, d.h. rechte *r* Spalten	linke Seite, d.h. linke *(r-1)* Spalten
voll iterativ	*iterativ*	*iterativ*
voll iterativ	*iterativ*	*separat*
voll iterativ	*separat*	*iterativ*
voll iterativ	*separat*	*separat*
separat	*iterativ*	*iterativ*
separat	*iterativ*	*separat*
separat	*separat*	*iterativ*
separat	*separat*	*separat*

In unserem Fall beinhaltet der Teil 1 die linke Seite der *r-GR*-Figur doppelt. Der minimale XOR-Aufwand der Ermittlung und der Addition dieser *(r-1)* Spalten der *r-GR*-Figur Nr.2 kann nicht kleiner als *(r-1)* sein, weil der Inhalt aller Spalten verschieden ist. Diese minimale Anzahl der XOR-Gatter kann mit der gleichen Art der Berechnung der linken Seiten beider *r-GR*-Figuren erreicht werden: einmal ermittelte Spalten-Werte (oder Spalten-Differenzen) der *r-GR*-Figur Nr.1 können weiterverwendet werden. Damit kann die Ermittlung der Spalten (oder Spalten-Differenzen) der rechten *r-GR*-Figur Nr.2 gespart werden. Ihr XOR-Aufwand besteht nur aus der Addition (XOR) ihrer Spalten (bzw. Spalten-Differenzen) mit dem Rest der GR, d.h. aus *(r-1)* XOR-Gatter.
Das begünstigt die Ablaufpläne, in denen gleiche Figuren-Teile gleichartig berechnet werden. **Tabelle 17** zeigt, welche Kombinationen der *iterativen* und *separaten* Berechnung der beiden Teile noch untersucht werden müssen.

[11] Kriterien ***(K)*** sind nur unter der Bedingung verwendbar, dass der *iterative* Sub-Teil für die Spalten $c^{n-GR}_{2n-2,}\ldots$, $c^{n-GR}_{2n-2-r+1}$ bereits existiert.

Tabelle 17: Untersuchte Ablaufpläne für n-GR-Gruppe 2

Teil 0	Teil 1			Bezeichnung des Ablaufplanes
	rest-GR-Figur Nr.1		*rest-GR*-Figur Nr.2	
	linke Seite	rechte Seite	linke Seite	
iterativ	*voll iterativ*	*iterativ*	*iterativ*	**A1**
voll iterativ	*separat*	*separat*	*separat*	**A2**
iterativ	*separat*	*iterativ*	*separat*	**A3**
voll iterativ	*voll iterativ*	*separat*	*iterativ*	**A4**

Wie es bereits oben erklärt wurde, ist für diese n-GR-Gruppe der XOR-Aufwand der optimalen Berechnung der linken *r* Spalten der *n*-GR gleich $\#\overset{optimal}{XOR}^{linke}_{\substack{Seite \\ r-GR-Figur}}$. Unter Berücksichtigung der Tatsache, dass die Berechnung der linken Seite der *r-GR*-Figur Nr.2 nach allen Ablaufplänen aus **Tabelle 17** gleich aufwendig ist (*r-1* XOR-Gatter), folgt, dass die Ablaufpläne **A2** und **A4** nur dann gleich aufwendig sind, wenn die *voll iterative* Berechnung der linken *r-GR*-Seite die optimale Berechnung ist. Sonst ist **A2** immer besser. Dasselbe gilt für **A1** und **A3**[12]. Damit bleiben nur zwei Ablaufpläne zum Vergleich, **A2** und **A3**.

Die Ermittlung des günstigsten Ablaufplanes und seines XOR-Aufwandes ist ausführlich im Anhang 1 dargelegt. Das Ergebnis der Analyse zeigt Formel **(126)**:

$$\#\overset{\text{optimal}}{XOR}_{c_{2n-2}^{n-GR}..c_n^{n-GR}} = \begin{cases} \#\overset{\text{A2}}{XOR}_{c_{2n-2}^{n-GR}..c_n^{n-GR}} = n + 2r - 5 + \underbrace{\#\overset{optimal}{XOR}^{linke}_{\substack{Seite \\ r-GR-Figur}}}_{=0,\ \text{wenn}\ 1 \le r \le 3}, \ 1 \le r \le 3 \\ \#\overset{\text{A3}}{XOR}_{c_{2n-2}^{n-GR}..c_n^{GR}} = \#\overset{optimal}{XOR}^{linke}_{\substack{Seite \\ r-GR-Figur}} + \#\overset{optimal}{XOR}_{c_{r-1}^{r-GR}} + n + r - 2, \ r \ge 4 \end{cases}$$

(126)

Die Ablaufpläne und ihre XOR-Aufwände für die *nicht vollen* GR bekommt man mit dem gleichen Verfahren wie für die volle GR. Teil 1 beinhaltet in dem Fall zwei *nicht volle NV-r-GR*-Figuren, weil die letzte Zeile – die Zeile mit dem TP p_n – leer ist. Auf die untersuchten Ablaufpläne hat diese Tatsache keine Auswirkung. Formel **(127)** stellt die Ergebnisse des Vergleiches der Ablaufpläne dar. Die Herleitung der Formel ist im Anhang 1 ausgeführt.

[12] Ablaufplan A3 ist besser, wenn $\#\overset{optimal}{XOR}_{c_{2r-2}^{r-GR}\ldots c_{r-1}^{r-GR}} \le \#\overset{voll\ iterativ}{XOR}_{c_{2r-2}^{r-GR}\ldots c_{r-1}^{r-GR}}$ gilt. Das hat aber keine Auswirkung auf die Auswahl der weiterhin betrachteten Ablaufpläne.

$$\#\ \overset{\text{optimal}}{\underset{\text{nicht volle GR}}{XOR}}_{c_{2n-3}^{NV-n-GR}\ldots c_n^{NV-n-GR}} = \begin{cases} \#\ \overset{A2}{XOR}_{c_{2n-3}^{NV-n-GR}\ldots c_n^{NV-n-GR}} = n-3,\ r \in \{1,2\} \\ \#\ \overset{A2}{XOR}_{c_{2n-3}^{NV-n-GR}\ldots c_n^{NV-n-GR}} = n-1,\ r = 3 \\ \#\ \overset{A3}{XOR}_{c_{2n-3}^{NV-n-GR}\ldots c_n^{NV-n-GR}} = n+r-4+ \\ \qquad +\#\ \underset{NV-r-GR-Figur}{\overset{optimal}{XOR}^{linke\ Seite}} \qquad +\#\ \overset{optimal}{XOR}_{c_{r-1}^{NV-r-GR}},\ r \geq 4 \end{cases} \quad \textbf{(127)}$$

***n*-GR-Gruppe 3**

Zu dieser Gruppe gehören alle Polynome, deren Länge $n=2^q+2^{q-1}+\ldots+2^{q-j}$, $0<j\leq q$ ist. **Abbildung 18** stellt die n-GR des typischen Vertreters der Gruppe 3 dar, 28-GR.

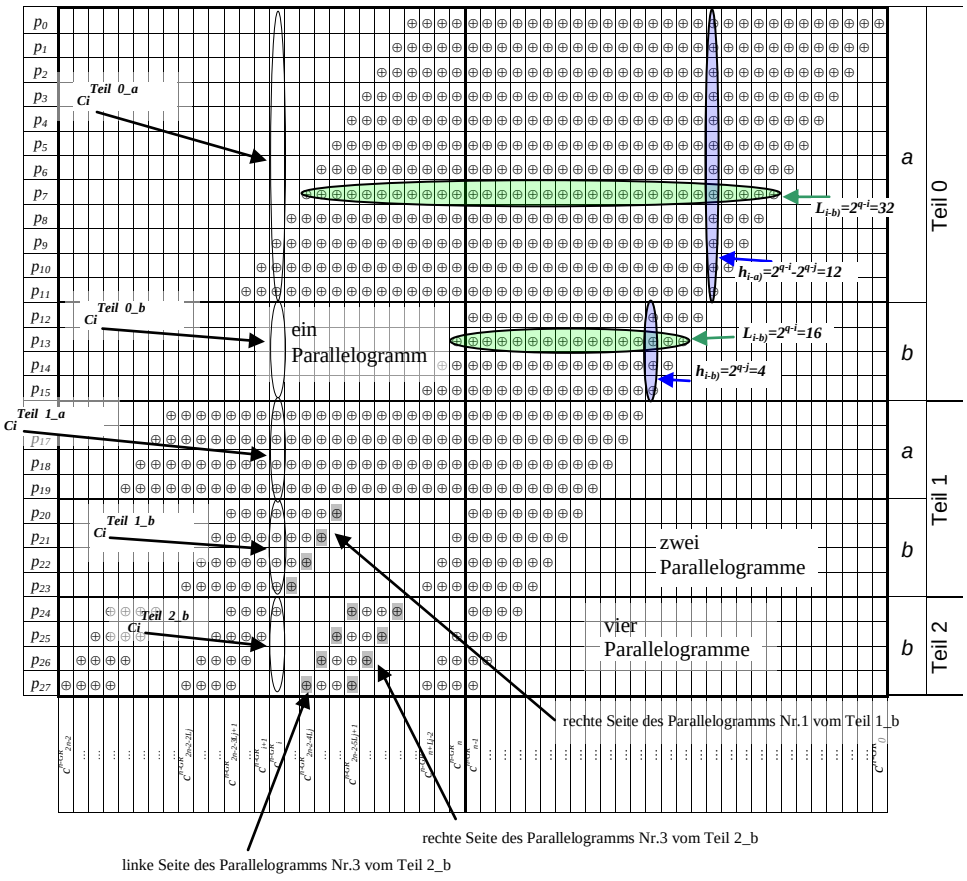


Abbildung 18: Vertreter der n-GR-Gruppe 3, *28-GR, $n=28=11100_2 \Rightarrow q=4, j=2$*

Die *n*-GR dieser Gruppe bestehen aus (*j+1*) folgenden Teilen:

• Jeder **Teil *i*** ($0 \le i \le j$) besteht aus 2^{q-i} Zeilen und kann in zwei weitere Sub-Teile folgendermaßen aufgeteilt werden:

- **Teil *i_a*** hat $(2^{q-i}-2^{q-j})$ Zeilen mit $L_{i\_a}=2^{q+1}$ ununterbrochen gefüllten Zellen, die ein Parallelogramm bilden
- **Teil *i_b*** besteht aus 2^{q-j} Zeilen, in denen 2^i gleiche Parallelogramme sich befinden. Diese Parallelogramme werden von links mit ‚1' beginnend durchnummeriert. Die Grundseite jedes Parallelogramms besteht aus $L_{i\_b}=2^{q-i}$ ununterbrochen gefüllter Zellen. Die dazugehörige Höhe entspricht $h_{i\_b}=2^{q-j}$ gefüllten Zellen. Die mittlere Spalte des rechtesten Parallelogramms befindet sich in der mittleren Spalte der GR. Der Abstand zwischen allen Parallelogrammen ist gleich und besteht aus folgender Anzahl leerer Zellen: $\frac{2^{q+1}-2^i \cdot 2^{q-i}}{2^i} = 2^{q-i}$

Bei dem **Teil *j*** besteht der **Teil *j_a*** aus $(2^{q-j}-2^{q-j})=0$ Zeilen, d.h. dass der **Teil *j*** nur den **Teil *j_b*** beinhaltet. Die Anzahl ununterbrochen gefüllter Zellen vom **Teil *j*** wird weiter mit L_j bezeichnet und die dazugehörige Höhe mit h_j.

Tabelle 18 erklärt die oben beschriebene Struktur der Aufteilung der n-GR.

Tabelle 18: Struktur der n-GR-Gruppe 3

n-GR-Teil *i*		TP	Anzahl der Zeilen im		Inhalt der Zeilen
i	*i*-Sub-Teil		*i*-Sub-Teil	Teil *i*	
0	***0_a***	$p_0 \dots p_2{}^{q}{}_{-2}{}^{q-j}{}_{-1}$	2^q-2^{q-j}	2^q	1Parallelogramm, $L_{0\_a}=2^{q+1}$, $h_{0\_a}=2^{q-j}-2^{q-j}$
	0_b	$p_2{}^{q}{}_{-2}{}^{q-j} \dots p_2{}^{q}{}_{-1}$	2^{q-j}		2^0=1Parallelogramm, $L_{0\_b}=2^q$, $h_{0\_b}=2^{q-j}$
1	***1_a***	$p_2{}^{q} \dots p_2{}^{q}{}_{+2}{}^{q-1}{}_{-2}{}^{q-j}{}_{-1}$	$2^{q-1}-2^{q-j}$	2^{q-1}	1Parallelogramm, $L_{1\_a}=2^{q+1}$, $h_{1\_a}=2^{q-1}-2^{q-j}$
	1_b	$p_2{}^{q}{}_{+2}{}^{q-1}{}_{-2}{}^{q-j} \dots p_2{}^{q}{}_{+2}{}^{q-1}{}_{-1}$	2^{q-j}		2^1=2 Parallelogramme, $L_{1\_b}=2^{q-1}$, $h_{1\_b}=2^{q-j}$
2	***2_a***	$p_2{}^{q}{}_{+2}{}^{q-1} \dots p_2{}^{q}{}_{+2}{}^{q-1}{}_{+2}{}^{q-2}{}_{-2}{}^{q-j}{}_{-1}$	$2^{q-2}-2^{q-j}$	2^{q-2}	1Parallelogramm, $L_{2\_a}=2^{q+1}$, $h_{2\_a}=2^{q-2}-2^{q-j}$
	2_b	$p_2{}^{q}{}_{+2}{}^{q-1}{}_{+2}{}^{q-2}{}_{-2}{}^{q-j} \dots p_2{}^{q}{}_{+2}{}^{q-1}{}_{+2}{}^{q-2}{}_{-1}$	2^{q-j}		2^2=4 Parallelogramme, $L_{2\_b}=2^{q-2}$, $h_{2\_b}=2^{q-j}$
…					
j	***j_a***	-	$2^{q-j}-2^{q-j}=0$	2^{q-j}	-
	j_b	$p_2{}^{q}{}_{+2}{}^{q-1}{}_{+\dots+2}{}^{q-j}{}_{-2}{}^{q-j} \dots p_2{}^{q}{}_{+2}{}^{q-1}{}_{+\dots+2}{}^{q-j}{}_{-1}$	2^{q-j}		2^j Parallelogramme, $L_j=2^{q-j}=h_j$

Für jeden **Teil *i_a***, $0 \le i<j$, gilt folgendes: jede Spalte dieses Teils beinhaltet in sich alle TP ihrer linken Nachbar-Spalte und kann ein weiteres Teil-Produkt beinhalten. Analog zur **(121)** ist hier die *iterative* (evtl. *voll iterative*) Berechnung optimal. Auf alle andere n-GR-Teile mit Ausnahme des Parallelogramms Nr.1 vom **Teil *j*** sind jetzt die Kriterien (***K***) anwendbar: alle GR-Teile mit *L>1* gehören zu dem *iterativen* TD-Teil.

Für alle Parallelogramme jedes **Teils *i_b***, $1 \le i<j$, gilt folgendes: Die linken Seiten aller Parallelogramme ab Nummer 2 befinden sich direkt unter den Seiten aller Parallelogramme des **Teils *(i-1)_b*** (siehe **Abbildung 18**). Diese Tatsache begünstigt den *iterativen* Ablaufplan der Berechnung, weil die Ermittlung der Spalten-Differenzen mithilfe bereits ermittelter Spalten-Differenzen besonders günstig berechnet werden kann.
Die Zuordnung des Parallelogramms Nr.1 vom **Teil *j*** zu dem *iterativen* oder *separaten* TD-Teil wird als Sonderfall betrachtet. Dies ist notwendig, da dieses Parallelogramm der einzige Inhalt der Spalten $c^{n\text{-}GR}_{2n\text{-}2}, \ldots, c^{n\text{-}GR}_{2n\text{-}2\text{-}Lj+1}$ ist[13]. Diese Tatsache begünstigt die *separate* Berechnung des Parallelogramms Nr.1 und damit die separate Berechnung des ganzen Teils *j.*
Die Zuordnung des ganzen Teils *j* wird im Fall $L_j=1$ auch als ein Sonderfall betrachtet. Nach Kriterien (***K***) soll er zum separaten TD-Teil zugeordnet werden. Unter Berücksichtigung der oben genannten Platzierung der Parallelogramm-Seiten muss jedoch auch die *iterative* Berechnung untersucht werden.
Tabelle 19 zeigt diese Sonderfälle und Ablaufpläne.

Tabelle 19: Untersuchte Ablaufpläne für n-GR-Gruppe 3

Fall	Teil *0_a…(j-1)_a*	Teil *(j-1)_b*	Teil *j*		Bezeichnung des Ablaufplanes
			Parallelogramm Nr.1	Parallelogramm Nr.2…Nr.2^j	
q-j≥2	*iterativ*	*iterativ*	*voll iterativ*	*iterativ*	**A1**
q-j=1	*voll iterativ*	*iterativ*	*separat*	*iterativ*	**A2**
q-j=0	*voll iterativ*	*iterativ*	*separat*	*separat*	**A3**
	voll iterativ	*separat* unter der Bedingung, dass die gefüllten Zellen von diesem Teil und vom *j.*Teil zusammen mehrere gleiche Figuren bilden	*separat*	*separat*	**A4**

Die Ablaufpläne **A1** und **A2** sind optimal für die jeweiligen Fälle. Im Fall *q=j* müssen die XOR-Aufwände der Ablaufpläne **A3** und **A4** verglichen werden, um den günstigsten von beiden zu bestimmen. Die Herleitung des XOR-Aufwandes entsprechend **Tabelle 19** ist im Anhang 1 gegeben. Formel **(128)** zeigt die Ergebnisse der Analyse:

$$\#^{\text{optimal}} XOR_{c^{n-GR}_{2n-2}\ldots c^{n-GR}_{n}} = \begin{cases} \#^{\text{A1}} XOR_{c^{n-GR}_{2n-2}\ldots c^{n-GR}_{n}} = \begin{cases} 2^{q+1}+2^{q-1}-3, & j=1,\ q\text{-}j\ge 2 \\ 2^{q+2}-3\cdot\left(2^{q-j}+1\right), & j>1,\ q\text{-}j\ge 2 \end{cases} \\ \#^{\text{A2}} XOR_{c^{n-GR}_{2n-2}\ldots c^{n-GR}_{n}} = \begin{cases} 3\cdot 2^{q}-7, & j=1,\ q\text{-}j=1 \\ 2^{q+2}-2^{q-j+1}-7, & j>1,\ q\text{-}j=1 \end{cases} \\ \#^{\text{A4}} XOR_{c^{n-GR}_{2n-2}\ldots c^{n-GR}_{n}} = 2^{q+2}-10+\begin{cases} 1, & j=2,\ j-q=0 \\ 0, & j\ne 2,\ j-q=0 \end{cases} \end{cases} \quad \textbf{(128)}$$

[13] Die *Kriterien (**K**)* sind nicht anwendbar.

Analog zur vollen n-GR ergeben sich die XOR-Aufwände und die günstigsten Ablaufpläne der Berechnung der nicht vollen GR (Herleitung siehe Anhang 1):

$$\overset{\text{optimal}}{\underset{\text{nicht volle GR}}{\#\ XOR}}_{c_{2n-3}^{NV-n-GR}\ldots c_n^{NV-n-GR}} = \begin{cases} \overset{A1}{\#\ XOR}_{c_{2n-3}^{NV-n-GR}\ldots c_n^{NV-n-GR}} = \begin{cases} 2^{q+1}+2^{q-1}-6, & j=1,\ q\text{-}j\geq 2 \\ 2^{q+2}-3\cdot 2^{q-j}-2^{j}-4, & j>1,\ q\text{-}j\geq 2 \end{cases} \\ \overset{A2}{\#\ XOR}_{c_{2n-3}^{NV-n-GR}\ldots c_n^{NV-n-GR}} = 7\cdot 2^{q-1}-11,\ q\text{-}j=1 \\ \overset{A1}{\#\ XOR}_{c_{2n-3}^{NV-n-GR}\ldots c_n^{NV-n-GR}} = 3\cdot 2^{q}-7,\ q\text{-}j=0 \end{cases} \quad \textbf{(129)}$$

n-GR-Gruppe 4

Zu dieser Gruppe gehören alle Polynome, deren Länge $n=2^q+2^{q-1}+\ldots+2^{q-j}+r$ ist, *mit* $j\neq 0,\ r\neq 0 \Rightarrow j\leq q-2$. **Abbildung 19** stellt den typischen Vertreter der n-GR-Gruppe 4 dar, 27-GR.

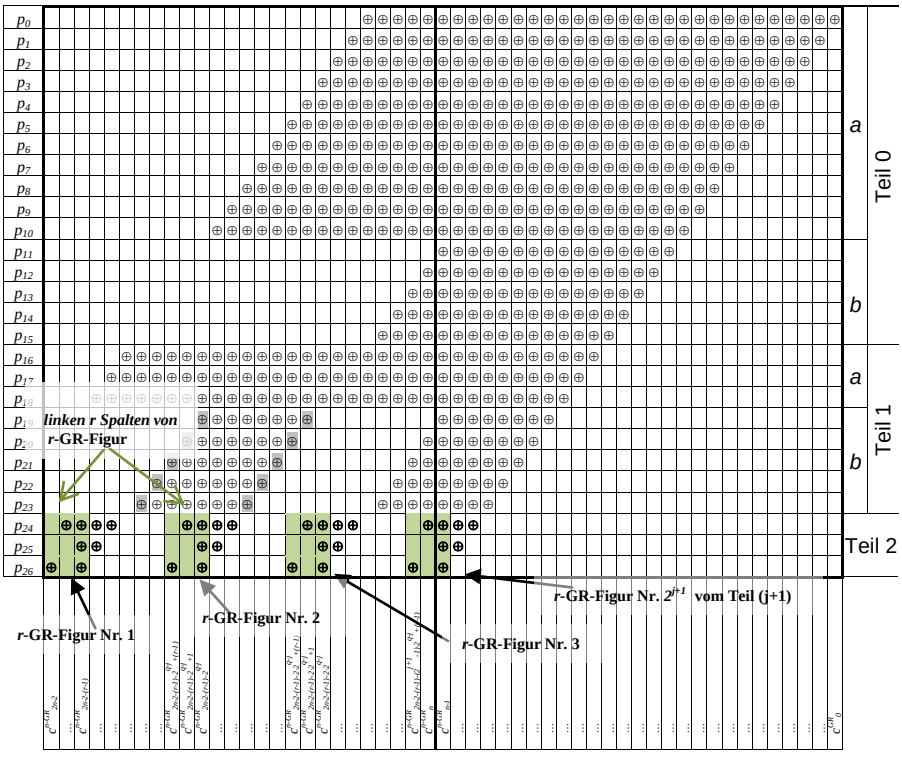


Abbildung 19: Vertreter der n-GR-Gruppe 4, 27-GR,
$n= 27=2^q+2^{q-1}+\ldots+2^{q-j}+r = 11011_2 \Rightarrow q=4,\ j=1$

Die n-GR dieser Gruppe bestehen aus (*j+2*) folgenden Teilen:

• Jeder **Teil *i***, $0 \le i \le j$, besteht aus 2^{q-i} Zeilen und kann in zwei Sub-Teile folgendermaßen aufgeteilt werden:

- **Teil *i_a*** besteht aus $n-\sum_{k=0}^{i} 2^{q-k}$ Zeilen mit $L=2^{q+1}$ ununterbrochen gefüllten Zellen
- **Teil *i_b*** besteht aus (2^{q-j}-*r*) Zeilen, in denen 2^i gleiche Parallelogramme platziert sind. Die Grundseite jedes Parallelogramms besteht aus $L=2^{q-i}$ ununterbrochen gefüllter Zellen. Die dazugehörige Höhe entspricht $h=2^{q-j}$-*r* gefüllter Zellen.

• Der **Teil *(j+1)*** besteht aus *r* Zeilen, die 2^{j+1} *r-GR*-Figuren beinhalten. Die *r-GR*-Figur Nr.1[14] befindet sich in den n-GR-Spalten $c^{n\text{-}GR}_{2n-2} \dots c^{n\text{-}GR}_{2n-2-(2r-1)}$. Die letzte *r-GR*-Figur (d.h. die mit der Nr. 2^{j+1}) befindet sich auf den n-GR-Spalten $c^{n\text{-}GR}_{n-1-r} \dots c^{GR}_{n-1+(r-1)}$. Ihre mittlere Spalte befindet sich in der mittleren Spalte der ganzen *n*-GR. Der Abstand zwischen allen *r-GR*-Figuren ist gleich $L=2^{q-(j+1)}$. Diese Zahl ist die Anzahl ununterbrochen gefüllter Zellen der Zeile mit dem TP $p_{2^q+\dots+2^{q-j}+1} = p_{n-r+1}$. **Tabelle 20** erklärt die oben beschriebene Struktur der Aufteilung der n-GR..

Tabelle 20: Struktur der n-GR-Gruppe 4

n-GR-Teil *i*		TP	Anzahl der Zeilen im		Inhalt der Zeilen
i	*i*-Sub-Teil		*i*-Sub-Teil	Teil *i*	
0	*0_a*	$p_0 \dots p_{n-2^q-1}$	$n-2^q$	2^q	$L_{0\text{-}a)}=2^{q+1}$ ununterbrochen gefüllte Zellen
	0_b	$p_{n-2^q} \dots p_{2^q-1}$	2^{q-j}-*r*		$2^0=1$Parallelogramm, $L_{0\text{-}b)}=2^q$, $h_{0\text{-}b)}=2^{q-j}$-*r*
1	*1_a*	$p_{2^q} \dots p_{2^q+(n-2^q-2^{q-1})-1}$	$n-2^q-2^{q-1}$	2^{q-1}	$L_{1\text{-}a)}=2^{q+1}$ ununterbrochen gefüllte Zellen
	1_b	$p_{2^q+(n-2^q-2^{q-1})} \dots p_{2^q+2^{q-1}-1}$	2^{q-j}-*r*		$2^1=2$ Parallelogramme, $L_{1\text{-}b)}=2^{q-1}$, $h_{1\text{-}b)}=2^{q-j}$-*r*
2	*2_a*	$p_{2^q+2^{q-1}} \dots p_{2^q+2^{q-1}+(n-2^q-2^{q-1}-2^{q-2})-1}$	$n-2^q-2^{q-1}-2^{q-2}$	2^{q-2}	$L_{2\text{-}a)}=2^{q+1}$ ununterbrochen gefüllte Zellen
	2_b	$p_{2^q+2^{q-1}+(n-2^q-2^{q-1}-2^{q-2})} \dots p_{2^q+2^{q-1}+2^{q-2}-1}$	2^{q-j}-*r*		$2^2=4$ Parallelogramme, $L_{2\text{-}b)}=2^{q-2}$, $h_{2\text{-}b)}=2^{q-j}$-*r*
…	…	…	…	…	…
j	*j_a*	$p_{2^q+2^{q-1}+\dots+2^{q-(j-1}1} \dots p_{2^q+2^{q-1}+\dots+2^{q-(j-1)}+r-1}$	*r*	2^{q-j}	$L_{j\text{-}a)}=2^{q+1}$ ununterbrochen gefüllte Zellen
	j_b	$p_{2^q+2^{q-1}+\dots+2^{q-(j-1)}+r} \dots p_{2^q+\dots+2^{q-j}-1}$	2^{q-j}-*r*		2^j Parallelogramme, $L_{j\text{-}b)}=2^{q-j}$, $h_{j\text{-}b)}=2^{q-j}$-*r*
j+1		$p_{2^q+\dots+2^{q-j}} \dots p_{2^q+\dots+2^{q-j}+r-1}$	*r*	*r*	2^{j+1} *r-GR*-Figuren. Die minimale Anzahl der *r-GR*-Figuren ist $2^{1+1}=4$, weil $j \neq 0$

Für alle **Teile *i_a***, $0 \le i \le j$, gilt folgendes: jede Spalte dieses Teils beinhaltet in sich alle TP ihrer linken Nachbar-Spalte und kann ein weiteres Teil-Produkt beinhalten.

[14] Alle Figuren wurden von links nach rechts nummeriert.

Analog zur **(121)** ist hier die *iterative* (evtl. *voll iterative*) Berechnung optimal. Auf alle anderen n-GR-Teile mit Ausnahme der *r-GR*-Figur Nr.1 vom **Teil *(j+1)*** sind jetzt die Kriterien (***K***) anwendbar.
Aus allen n-GR-Teilen hat der **Teil *j*** die kleinste Anzahl ununterbrochen gefüllter Zellen: $L_{0-b)} > L_{1-b)} > \ldots > L_{(j-1)-b)} > L_{j-b)} = 2^{q-j}$, wobei $min(2^{q-j}) = 2^{q-(q-2)} = 4$. Nach den Kriterien ***(K)*** gehören alle **Teile *i_b***, $0 \le i \le j$, zu dem *iterativen* TD-Teil.
Für alle Parallelogramme jedes **Teils *i_b***, $1 \le i \le j$, gilt folgendes: Die linken Seiten aller Parallelogramme ab Nr. 2 befinden sich direkt unter den Seiten aller Parallelogramme des **Teils *(i-1)_b*** (siehe **Abbildung 19**). Dasselbe gilt auch für alle *r-GR*-Figuren ab *r-GR*-Figur Nr.2: Die linken *r* Spalten dieser Figuren befinden sich direkt unter den Seiten der Parallelogramme aus dem **Teil *j_b*** (siehe **Abbildung 19**). Diese Position der Parallelogramme und der *r-GR*-Figuren begünstigt die *iterative* Berechnung der ganzen linken *n-GR*-Seite.
Anderseits sind die linken *r* Spalten der *r-GR*-Figur Nr.1 der einzige Inhalt der entsprechenden Spalten der ganzen n-GR, was die *separate* Berechnung des **Teils *(j+1)*** begünstigt. Die Zuordnung der *r-GR*-Figur Nr.1 wurde im Anhang 1 analysiert. Aus der Analyse ergeben sich die folgenden Ablaufpläne der Berechnung der linken n-GR-Seite (siehe **Tabelle 21**) als die optimalen:

Tabelle 21: Optimale Ablaufpläne für n-GR-Gruppe 4

Fall	Teil *i*, $1 \le i \le j$	Teil *(j+1)*							Bezeichnung des Ablaufplanes
		r-GR-Figur Nr.1		*r-GR*-Figur Nr.2		*r-GR*-Figur Nr.3		…	*r-GR*-Figur Nr.2^j	
		linke Seite	rechte Seite	linke Seite	rechte Seite	linke Seite	rechte Seite		linke Seite, *(r-1) Spalten*	
r=1	*voll iterativ*	*separat*	*separat*	*separat*	*separat*	*separat*		…	*separat*	**A4**
$r \in \{2,3\}$	*voll iterativ*	*separat*	*separat*	*iterativ, (r-1)* Spalten	*separat, r* Spalten	siehe *r-GR*-Figur Nr.2		…	siehe *r-GR*-Figur Nr.2	**A3-c**
r≥4, gerade	*iterativ*	*voll iterativ*	*iterativ*	*iterativ*	*iterativ*	siehe *r-GR*-Figur Nr.2		…	siehe *r-GR*-Figur Nr.2	**A1**
r≥4, ungerade	*iterativ*	$c^{n\text{-}GR}_{2n-2} = c^{r\text{-}GR}_{2r-2}$ *separat, ab* $c^{n\text{-}GR}_{2n-3}$ *voll iterativ*	*iterativ,* $c^{n\text{-}GR}_{2n-r} = c^{r\text{-}GR}_{r+1}$	*iterativ*	*iterativ*	siehe *r-GR*-Figur Nr.2		…	siehe *r-GR*-Figur Nr.2	**A1-b**

Die XOR-Aufwände der Ablaufplänen aus **Tabelle 21** können folgendermaßen in einer Formel angegeben werden:

$$\#^{optimal}XOR_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}} = \begin{cases} \#^{A4}XOR_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}} = 2n-2^{q-j}-5, r=1 \\ \#^{A3-c}XOR_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}} = 2n-2^{q-j}-3, r=2 \\ \#^{A3-c}XOR_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}} = 2n-2^{q-j}-1, r=3 \\ \#^{A1}XOR_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}} = 2n-2^{q-j}-2+ \\ \quad +\#^{voll\ iterativ}XOR_{\substack{linke\ Seite \\ r-GR-Figur}} +\#^{voll\ iterativ}XOR_{c_{r-1}^{r-GR}}, \text{gerades } r \geq 4 \\ \#^{A1-b}XOR_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}} = 2n-2^{q-j}-3+ \\ \quad +\#^{voll\ iterativ}XOR_{\substack{linke\ Seite \\ r-GR-Figur}} +\#^{voll\ iterativ}XOR_{c_{r-1}^{r-GR}}, \text{ungerades } r \geq 4 \end{cases}$$

(130)

Tabelle 22 zeigt die günstigsten Ablaufpläne der Berechnung der *nicht vollen* GR dieser n-GR-Gruppe. Für *r≥4* ist der optimale Ablaufplan zu **A3** aus **Tabelle 21** identisch. Für alle *1<r≤3* ist ein anderer Ablaufplan, der **A1-*NV***, optimal.

Tabelle 22: optimierte Ablaufpläne der Berechnung der *NV-n*-GR

Fall	Teil *i, 1≤i≤j*	Teil *(j+1)*								Bezeichnung des Ablaufplane
		r-NV-GR-Figur Nr.1		*r-NV-GR*-Figur Nr.2		*r-NV-GR*-Figur Nr.3		…	*r-NV-GR*-Figur Nr.2^j	
		linke Seite	rechte Seite	linke Seite	rechte Seite	linke Seite	rechte Seite		linke Seite, *(r-1) Spalten*	
r=1	*voll iterativ*	-	-	-	-	-		…	-	**A1**
1<r≤3	*voll iterativ*	*separat*	*separat*	*iterativ*	*iterativ*	siehe *r-NV-GR*-Figur Nr.2		…	siehe *r-NV-GR*-Figur Nr.2	**A1-NV**
r≥4	*iterativ*	*voll iterativ*	*iterativ*	*iterativ*	*iterativ*	siehe *r-NV-GR*-Figur Nr.2		…	siehe *r-NV-GR*-Figur Nr.2	**A1**

Die XOR-Aufwände der Ablaufpläne aus **Tabelle 22** sind in Formel **(131)** angegeben:

$$\#^{\substack{optimal \\ nicht\ volle\ GR}}XOR_{c_{2n-3}^{NV-n-GR}\ldots c_n^{NV-n-GR}} = \begin{cases} \#^{A1}XOR_{c_{2n-3}^{NV-n-GR}\ldots c_n^{NV-n-GR}} = 2n-2^{q-j}-2^{j+1}-3,\ r=1 \\ \#^{A1\text{-}NV}XOR_{c_{2n-3}^{NV-n-GR}\ldots c_n^{NV-n-GR}} = 2n-2^{q-j}-2^{j+1}-3,\ r=2 \\ \#^{A1\text{-}NV}XOR_{c_{2n-3}^{NV-n-GR}\ldots c_n^{NV-n-GR}} = 2n-2^{q-j}-2^{j+1}-2,\ r=3 \\ \#^{A1}XOR_{c_{2n-3}^{NV-n-GR}\ldots c_n^{NV-n-GR}} = 2n-2^{q-j}-2^{j+1}-2+ \\ \quad +\#^{voll\ iterativ}XOR_{\substack{linke\ Seite \\ NV-r-GR-Figur}} +\#^{voll\ iterativ}XOR_{c_{r-1}^{NV-r-GR}},\ r \geq 4 \end{cases}$$

(131)

Die XOR-Aufwände der Berechnung der ganzen n-GR aller vier n-GR-Gruppen können für jede Polynom-Länge *n* (für alle *n>3*, weil der optimale Ablaufplan und sein XOR-Aufwand der Berechnung der linken Seite der 3-GR eindeutig sind) folgendermaßen in einer Formel vereinigt werden:

$$\#XOR_{n-GR}^{\text{optimal}} = 2n-3+\begin{cases} 0,\ r=0,\ j=0 \\ 2r-3,\ 1\le r\le 3,\ j=0 \\ r+\#XOR_{\substack{\text{linke Seite} \\ r-GR-Figur}}^{\text{optimal}}+\#XOR_{c_{r-1}^{r\text{-}GR}}^{optimal},\ r\ge 4,\ j=0 \\ 2^q-1,\ r=0,\ j=1\le q\text{-}2 \\ n-2^{q-j}-1,\ r=0,\ 1<j\le q\text{-}2 \\ 1, r=0,\ j=1=q-1 \\ n-5,\ r=0,\ 1<j=q-1 \\ -1,\ r=0,\ 1=j=q \\ 2,\ r=0, 2=j=q \\ n-6,\ r=0, 2<j=q \\ n-2^{q-j}+2r-5,\ 1\le r\le 3,\ j>0 \\ n-2^{q-j}+\#XOR_{\substack{\text{linke Seite} \\ r-GR-Figur}}^{voll\ iterativ}+ \\ \quad +\#XOR_{c_{r-1}^{r\text{-}GR}}^{voll\ iterativ},\ \text{gerades}\ \ r\ge 4,\ j>0 \\ n-2^{q-j}-1+\#XOR_{\substack{\text{linke Seite} \\ r-GR-Figur}}^{voll\ iterativ}+ \\ \quad +\#XOR_{c_{r-1}^{r\text{-}GR}}^{voll\ iterativ},\ \text{ungerades}\ \ r\ge 4,\ j>0 \end{cases} \quad \textbf{(132)}$$

$$\#\ \overset{\text{optimal}}{\underset{\text{nicht volle GR}}{XOR}}_{NV-n-GR} = 2n-5+\begin{cases} 0,\ r\in\{0,1,2\},\ j=0 \\ 2,\ r=3,\ j=0 \\ r-1+\#\ \overset{optimal}{XOR}^{linke\ Seite}_{NV-r-GR-Figur}+ \\ \quad +\#\ \overset{optimal}{XOR}_{c_{r-1}^{NV-r-GR}},\ r\geq 4,\ j=0 \\ \\ 2^q-3,\ r=0,\ 1=j\leq q\text{-}2 \\ n-2^{q-j}-2^j-1,\ r=0,\ 1<j\leq q\text{-}2 \\ n-2^{q-1}-4,\ r=0,\ j=q\text{-}1>0 \\ 0,\ r=0,\ j=q=1 \\ n-2^q-2,\ r=0,\ j=q>1 \\ n-2^{q-j}-2^{j+1},\ r\in\{1,2\},\ j>0 \\ n-2^{q-j}-2^{j+1}+1,\ r=3,\ j>0 \\ \\ n-2^{q-j}-2^{j+1}+1+\#\ \overset{voll\ iterativ}{XOR}^{linke\ Seite}_{NV-r-GR-Figur}+ \\ \quad +\#\ \overset{voll\ iterativ}{XOR}_{c_{r-1}^{NV-r-GR}},\ r\geq 4,\ j>0 \end{cases} \qquad \textbf{(133)}$$

Formeln **(132)** und **(133)** können für die Ermittlung des XOR-Aufwandes der n-GR und der NV-n-GR algorithmisch (siehe **Algorithmus 3**) nur dann verwendet werden, wenn die XOR-Aufwände der Berechnung der linken Seiten aller *i*-GR, *NV-i*-GR, und der *i*-GR-Spalte c_{i-1}, $i<n$, nach *optimalen* und nach *voll iterativen* Ablaufplänen zur Verfügung stehen. Die Formeln **(134)**-**(137)** beinhalten diese Daten. Die Herleitung dieser Formeln ist im Anhang 1 gegeben.

$$\#\overset{\text{optimal}}{XOR}_{i-GR}^{linkeSeite} = i-2+\begin{cases} 0,\ r=0,\ j=0 \\ 2r-3, 1\le r\le 3,\ j=0 \\ r+\#\overset{\text{optimal}}{XOR}_{\substack{linke\\ Seite\\ r-GR-Figur}} +\#\overset{optimal}{XOR}_{C_{r-1}^{r-GR}},\ r\ge 4,\ j=0 \\ 2^q-1,\ r=0,\ j=1\le q\text{-}2 \\ i-2^{q-j}-1, r=0, 1<j\le q\text{-}2 \\ 1, r=0,\ j=1=q-1 \\ i-5,\ r=0, 1<j=q-1 \\ -1, r=0, 1=j=q \\ 2, r=0, 2=j=q \\ i-6,\ r=0, 2<j=q \\ i-2^{q-j}+2r-5, 1\le r\le 3,\ j>0 \\ i-2^{q-j}+\#\overset{voll\ iterativ}{XOR}_{\substack{linke\\ Seite\\ r-GR-Figur}} +\#\overset{voll\ iterativ}{XOR}_{C_{r-1}^{r-GR}},\ \text{gerades}\ r\ge 4,\ j>0 \\ i-2^{q-j}-1+\#\overset{voll\ iterativ}{XOR}_{\substack{linke\\ Seite\\ r-GR-Figur}} +\#\overset{voll\ iterativ}{XOR}_{C_{r-1}^{r-GR}}, \\ \qquad\qquad \text{ungerades}\ r\ge 4,\ j>0 \end{cases}$$

$$\#\overset{optimal}{XOR}_{C_{i-1}^{i-GR}} = \begin{cases} 1,\ r=0,\ j=0 \\ 2,\ r>0,\ j=0 \\ 2,\ r=0,\ j>0 \,\middle|\, j\ne q\ge 3 \\ 3,\ r=0,\ j=q\ge 3 \\ 3,\ r\in\{1,2,3\},\ j>0 \\ 2,\ r>3,\ j>0 \end{cases}$$

(134)

$$\#\ XOR^{voll\ iterativ}_{i-GR}{}^{linkeSeite} = i-2+\begin{cases} 0,\ r=0,\ j=0 \\ -r+2,\ r=1,\ j=0 \\ \#\ XOR^{voll\ iterativ}_{r-GR-Figur}{}^{linke\ Seite} + \#\ XOR^{voll\ iterativ}_{c_{r-1}^{r-GR}} + r,\ r>1,\ j=0 \\ 2^q-1,\ r=0,\ j=1 \\ i-2^{q-j}-1,\ r=0,\ j>1 \\ i-2^{q-j},\ \ r=1,\ j>0 \\ i-2^{q-j} + \#\ XOR^{voll\ iterativ}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#\ XOR^{voll\ iterativ}_{c_{r-1}^{r-GR}},\ \ r>1,\ j>0 \end{cases}$$

$$\#\ XOR^{voll\ iterativ}_{c_{i-1}^{i-GR}} = \begin{cases} 1,\ r=0,\ j=0 \\ 2,\ r>0,\ j=0 \\ 2, r=0,\ j>0 \\ 2,\ r>0,\ j>0 \end{cases}$$

(135)

$$\#\ XOR^{\text{optimal}}_{\substack{\text{nicht volle GR}\\ NV-i-GR}}{}^{linkeSeite} = 2i-3+\begin{cases} 0,\ r\in\{0,1,2\},\ j=0 \\ 2,\ r=3,\ j=0 \\ r-1+\#\ XOR^{optimal\ linke\ Seite}_{NV-r-GR-Figur}+\#\ XOR^{optimal}_{c_{r-1}^{NV-r-GR}},\ r\geq 4,\ j=0 \\ 2^q-3,\ r=0,\ \ 1=j\leq q\text{-}2 \\ i-2^{q-j}-2^j-1,\ r=0,\ \ 1<j\leq q\text{-}2 \\ i-2^{q-1}-4,\ r=0,\ \ j=q\text{-}1>0 \\ 0,\ r=0,\ j=q=1 \\ i-2^q-2,\ r=0,\ j=q>1 \\ i-2^{q-j}-2^{j+1},\ r\in\{1,2\},\ j>0 \\ i-2^{q-j}-2^{j+1}+1,\ r=3,\ j>0 \\ i-2^{q-j}-2^{j+1}+1+\#\ XOR^{voll\ iterativ\ linke\ Seite}_{NV-r-GR-Figur}+ \\ \quad +\#\ XOR^{voll\ iterativ}_{c_{r-1}^{NV-r-GR}},r\geq 4,\ j>0 \end{cases}$$

$$\#\ XOR^{optimal}_{c_{i-1}^{NV-i-GR}} = \begin{cases} 1,\ r=0,\ j=0 \\ 1,\ r=1,\ j=0 \\ 2,\ r>1,\ j=0 \\ 1,\ r=0,\ j=q=1 \\ 2,\ r=0,\ j>0 \,|\, j\neq q=1 \\ 2,\ r>0,\ j>0 \end{cases}$$

(136)

$$\#\ \overset{voll\ iterativ}{XOR}\ {}^{linkeSeite}_{NV-i-GR} = i-3+\begin{cases} 0,\ r=0,\ j=0 \\ 0,\ r=1,\ j=0 \\ r-1+\#\ \overset{voll\ iterativ}{XOR}\ {}^{linke\ Seite}_{NV-r-GR-Figur}+ \\ \quad +\ \overset{voll\ iterativ}{XOR}\ _{c_{r-1}^{NV-r-GR}},\ r>1,\ j=0 \\ 2^q-3,\ r=0,\ j=1 \\ i-2^{q-j}-2^j-1,\ r=0,\ j>1 \\ i-2^{q-j}-2^{j+1},\ r=1,\ j>0 \\ i-2^{q-j}-2^{j+1}+1+\#\ \overset{voll\ iterativ}{XOR}_{c_{2r-3}^{NV-r-GR}\ldots c_r^{NV-r-GR}}+ \\ \quad +\ \#\ \overset{voll\ iterativ}{XOR}_{c_{r-1}^{NV-r-GR}},\ r>1,\ j>0 \end{cases}$$

$$\#\ \overset{voll\ iterativ}{XOR}\ _{c_{i-1}^{NV-i-GR}} = \begin{cases} 1,\ r=0,\ j=0 \\ 1,\ r=1,\ j=0 \\ 2,\ r>1,\ j=0 \\ 2,\ r=0,\ j>0 \\ 2,\ r>0,\ j>0 \end{cases}$$

(137)

Die Gatter-Komplexität der *iterativ optimierten* Karatsuba-MM für *n*-Bits Polynome wurde mittels des ***Algorithmus 3*** und Formeln **(132)** – **(137)** für alle *n≤600* ermittelt und als eine Tabelle, die für den ***Algorithmus 2*** zur Verfügung steht, gespeichert.

4.3.5. Iterative Optimierung der Karatsuba-n-Segment-TD

In dieser Arbeit wurde die Berechnung der Karatsuba-*n*-Segment-TD, 2≤*n*≤*16*, iterativ optimiert. Die Karatsuba-*n*-Segment-TD können aus Karatsuba-*n*-TD ermittelt werden (siehe Kapitel 3.6). In diesem Kapitel wird die Ermittlung der GK der *iterativ optimierten* Berechnung der Karatsuba-*n*-Segment-TD an dem Beispiel der 4-Segment-TD erklärt. **Tabelle 23** zeigt die GK für alle untersuchten und im ***Algorithmus 2*** benutzten Segmentierungen. Nach der Analyse der Ergebnisse des ***Algorithmus 2*** wurde festgestellt, dass nur die Aufteilung der Polynome in n≤7 Segmente eine praktische Bedeutung hat.

Tabelle 23: Gatter-Komplexität
Karatsuba-*n*-Segment-TD, mit $2 \le n \le 16$

n	*Gatter-Komplexität*	*delay*, T_{XOR}
2	$GC_{2m}=3 \cdot GC_m+(7m-3)_{XOR}$	3
3	$GC_{3m}=6 \cdot GC_m+(21m-8)_{XOR}$	4
4	$GC_{4m}=9 \cdot GC_m+(34m-11)_{XOR}$	5
5	$GC_{5m}=14 \cdot GC_m+(65m-22)_{XOR}$	6
6	$GC_{6m}=18 \cdot GC_m+(85m-27)_{XOR}$	7
7	$GC_{7m}=23 \cdot GC_m+(119m-39)_{XOR}$	8
8	$GC_{8m}=27 \cdot GC_m+(133m-41)_{XOR}$	9
9	$GC_{9m}=36 \cdot GC_m+(205m-75)_{XOR}$	10
10	$GC_{10m}=42 \cdot GC_m+(233m-75)_{XOR}$	11
11	$GC_{11m}=49 \cdot GC_m+(287m-98)_{XOR}$	12
12	$GC_{12m}=54 \cdot GC_m+(296m-85)_{XOR}$	13
13	$GC_{13m}=63 \cdot GC_m+(388m-131)_{XOR}$	14
14	$GC_{14m}=69 \cdot GC_m+(421m-135)_{XOR}$	15
15	$GC_{15m}=76 \cdot GC_m+(458m-144)_{XOR}$	16
16	$GC_{16m}=81 \cdot GC_m+(469m-144)_{XOR}$	17

Nun wird die GK der iterativ optimierten Berechnung der Karatsuba-4-Segment-TD ermittelt. **Tabelle 24** ist die Karatsuba-4-Segment-TD. Alle höchstwertigen, (m-1)-Bits große, TP-Segmente sind grau gekennzeichnet. Alle niedrigstwertigen Segmente sind m-Bits groß.

Tabelle 24: Karatsuba-4-Segment-TD

	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0
$A_0 \cdot B_0[0]=TP_1[0]$					⊕	⊕	⊕	⊕
$A_0 \cdot B_0[1]=TP_1[1]$				⊕	⊕	⊕	⊕	
$A_1 \cdot B_1[0]=TP_2[0]$				⊕	⊕	⊕	⊕	
$A_1 \cdot B_1[1]=TP_2[1]$			⊕	⊕	⊕	⊕		
$A_2 \cdot B_2[0]=TP_3[0]$			⊕	⊕	⊕	⊕		
$A_2 \cdot B_2[1]=TP_3[1]$		⊕	⊕	⊕	⊕			
$A_3 \cdot B_3[0]=TP_4[0]$		⊕	⊕	⊕	⊕			
$A_3 \cdot B_3[1]=TP_4[1]$	⊕	⊕	⊕	⊕				
$(A_0 \oplus A_2) \cdot (B_0 \oplus B_2)[0]=TP_5[0]$					⊕	⊕		
$(A_0 \oplus A_2) \cdot (B_0 \oplus B_2)[1]=TP_5[1]$				⊕	⊕			
$(A_1 \oplus A_3) \cdot (B_1 \oplus B_3)[0]=TP_6[0]$				⊕	⊕			
$(A_1 \oplus A_3) \cdot (B_1 \oplus B_3)[1]=TP_6[1]$			⊕	⊕				
$(A_0 \oplus A_1) \cdot (B_0 \oplus B_1)[0]=TP_7[0]$					⊕		⊕	
$(A_0 \oplus A_1) \cdot (B_0 \oplus B_1)[1]=TP_7[1]$				⊕		⊕		
$(A_2 \oplus A_3) \cdot (B_2 \oplus B_3)[0]=TP_8[0]$			⊕		⊕			
$(A_2 \oplus A_3) \cdot (B_2 \oplus B_3)[1]=TP_8[1]$		⊕		⊕				
$(A_0 \oplus A_1 \oplus A_2 \oplus A_3) \cdot (B_0 \oplus B_1 \oplus B_2 \oplus B_3)[0]=TP_9[0]$					⊕			
$(A_0 \oplus A_1 \oplus A_2 \oplus A_3) \cdot (B_0 \oplus B_1 \oplus B_2 \oplus B_3)[1]=TP_9[1]$				⊕				

Tabelle 25 zeigt die Aufteilung der Karatsuba-4-Segment-TD (siehe **Tabelle 24**) in einen *iterativ* und einen *separat* zu berechnenden TD-Teil, wobei bei dem *iterativen*

TD-Teil die gleichgefüllten Zeilen bereits zusammen geXORt (Zeilen-Implosion) wurden.

Tabelle 25: Aufteilung der Karatsuba-4-Segment-TD

$TP_1[0]$					⊕	⊕	⊕	⊕	iterativer TD-Teil (Sub-TD 1)
$TP_1[1] \oplus TP_2[0]$				⊕	⊕	⊕	⊕		
$TP_2[1] \oplus TP_3[0]$			⊕	⊕	⊕	⊕			
$TP_3[1] \oplus TP_4[0]$		⊕	⊕	⊕	⊕				
$TP_4[1]$	⊕	⊕	⊕	⊕					
$TP_5[0]$					⊕	⊕			
$TP_5[1] \oplus TP_6[0]$				⊕	⊕				
$TP_6[1]$			⊕	⊕					
$TP_7[0]$					⊕		⊕		separater TD-Teil (Sub-TD 2)
$TP_8[0]$			⊕		⊕				
$TP_9[0]$					⊕				
$TP_7[1]$				⊕		⊕			
$TP_8[1]$		⊕		⊕					
$TP_9[1]$				⊕					
	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0	

Der XOR-Aufwand der Berechnung der *n*-Segment-TD besteht aus:

- Berechnung der TM-Eingänge: $2m \cdot \underbrace{5}_{TP_5,\cdots,TP_9} = 10m \cdot$
- Durchführung der Zeilen-Implosion des *iterativen* TD-Teil: 4(*m*-1)
- Berechnung des *iterativen* TD-Teils nach dem Ablaufplan $C^1_7 \rightarrow C^1_6 \rightarrow C^1_5 \rightarrow C^1_4$ und $C^1_0 \rightarrow C^1_1 \rightarrow C^1_2 \rightarrow C^1_3$:

$$\underbrace{(m-1)+(2m-1)+2m}_{\text{linke } n \text{ Spalten}} + \underbrace{m+2m+2m}_{\text{rechte } n \text{ Spalten}} = 10m-2$$

- Addition (XOR) der TP-Segmente von dem *separaten* TD-Teil:

$$5m+5(m-1) = 10m-5$$

Die gesamte Anzahl der XOR- Gatter ist:

$$\#XOR = \underbrace{10m}_{\substack{\text{Berechnung} \\ \text{TM-Operanden}}} + \underbrace{4m-4+10m-2}_{\substack{\text{Berechnung } \textit{iterativer} \text{ Sub-TD,} \\ \text{einschließlich Zeilen-Implosion}}} + \underbrace{10m-5}_{\substack{\text{XOR der TP von } \textit{separater} \text{ Sub-TD}}} = 34m-11$$

inclusive

So ist die Gatter-Komplexität der *n*-Segment Polynom-Multiplikation nach der iterativ optimierten Karatsuba-Multiplikations-Methode:

$$GC_{4m}^{\substack{\textit{iterativ optimierte} \\ \textit{Karatsuba-4-Segment-MM}}} = 9 \cdot GC_{m-TM} + (34m-11)_{XOR}$$

(138)

Die GC des längsten Pfades, die die Signalverzögerung *delay* der Karatsuba-4-Segment-TD verursacht, bezieht sich auf die Berechnung der rechten TD-Seite und auf die Addition (XOR) der TP-Segmente vom separaten TD-Teil. Formel **(139)** stellt die GC des LP und die verursachte Signalverzögerung dar.

$$GC_{\substack{LP\\4m}} = 1\cdot \overset{\text{MM des } m\text{-TM}}{\text{GC}}_{LP \text{ des } m\text{-TM}} + \left(\underbrace{4}_{\substack{iterativer\\ TD-Teil}} + \underbrace{1}_{\substack{\text{separater}\\ \text{TD-Teil}}} \right)_{XOR} = \overset{\text{MM des } m\text{-TM}}{\text{GC}}_{LP \text{ des } m\text{-TM}} + 5_{XOR}$$

$$delay_{4m} = \overset{\text{MM des } m\text{-TM}}{delay}_{m\text{-TM}} + 5\cdot T_{XOR}$$

(139)

4.4. Herleitung und iterative Optimierung der Winograd-MM für n-Term-Polynome

Ähnlich wie bei der Karatsuba-n-Segment-Multiplikation können die iterativ optimierten Ablaufpläne der Winograd-MM für n-Bits- und n-Segment-großen Polynome ermittelt werden. Die Berechnung der exakten GK für Winograd-*n*-TD wird für *n≤600* und für Winograd-*n-Segment*-TD nur für die Segmentierung der Operanden in *n* Teile, *2≤n≤16*, durchgeführt.
Nach der Analyse der Ergebnisse des ***Algorithmus 2*** wurde festgestellt, dass die Segmentierung der Polynome nur in 3, 6 und 9 Teile eine praktische Bedeutung hat.

4.4.1. Besonderheit der Winograd-n-TD

In diesem Kapitel wird die GK der *iterativ optimierten* Berechnung der Winograd-MM als eine Funktion von der Länge der Polynome *n* ermittelt. Dafür wurden die TD der Winograd-MF für *n*-Bits Polynome, *1≤n≤243*, analysiert. Für jedes *n* sind der Produkt-Ausdruck und seine TD einzigartig. Alle n-TD wurden aus der Winograd-243-TD auf die im Kapitel 4.3.1 beschriebene Art ermittelt[15]. Um die MF der 243-Bits Polynome ($243=3^5$) zu ermitteln, wurde Winograd-MF für 3-Segment Polynome **(140)** mehrmals rekursiv angewendet.

$$\begin{aligned} A\cdot B = A_2A_1A_0\cdot B_2B_1B_0 &= A_2B_2\cdot 2^{4m} \oplus \\ &\oplus \left(\left(A_2\oplus A_1\right)\left(B_2\oplus B_1\right)\oplus A_1B_1\oplus A_2B_2\right)\cdot 2^{3m} \oplus \\ &\oplus \left(\left(A_2\oplus A_0\right)\left(B_2\oplus B_0\right)\oplus A_0B_0\oplus A_2B_2\oplus A_1B_1\right)\cdot 2^{2m} \oplus \\ &\oplus \left(\left(A_0\oplus A_1\right)\left(B_0\oplus B_1\right)\oplus A_1B_1\oplus A_0B_0\right)\cdot 2^{m} \oplus A_0B_0\cdot 2^{0} \end{aligned}$$

(140)

[15] Aus der 243-TD wurden die TD aller kleineren Polynom-Längen *n* mit dem Setzen der höchstwertigen Bits der beiden Multiplikanden zu 0 abgeleitet. Dies ist möglich, weil jede *n*-Bits große Zahl mit höchstwertigem Bit 0 bereits (*n-1*)-Bits groß ist.

Die Erstellung der *n*-TD der Winograd-MF mit dieser Art der Ableitung wurde für $n \leq 243$ programmiert.

Bei der Winograd-*n*-TD können die gleichen Besonderheiten wie bei der Karatsuba-*n*-TD festgestellt werden:

- jede *n*-TD besteht aus der n-GR und weiteren Sub-TD, die der *i*- oder i^{NV}-TD identisch sind, mit $i<n$. **Tabelle 26** zeigt, in welche Sub-TD eine *n*-TD ($n \leq 27$) aufgeteilt werden kann
- die Zahlen in der **Tabelle 26** bilden „Strahlen". Alle Sub-TD des rechten Strahls (Strahl 1) haben $L=3^0=1$ ununterbrochen gefüllte Zellen. Jeder Strahl links vom Strahl 1, der Strahl *i*, refenziert die Sub-TD mit $L=3^{i-1}>2$ ununterbrochen gefüllten Zellen. Nach Kriterien ***(K)*** gehören alle Sub-TD des Strahls 1 zu dem *separaten* und alle anderen Sub-TD zu dem *iterativen* Sub-Teil der TD, wenn die GR den *iterativen* Sub-Teil der TD für alle TD-Spalten bildet.
- **Tabelle 26** kann für beliebig große *n* ohne Kenntnisse der *n*-TD erweitert werden. Der Erweiterungsprozess wurde in C programmiert. Die Korrektheit der entstehenden Tabelle wurde für alle *n*-TD mit $27<n \leq 243$ geprüft.

Tabelle 26: Aufteilung der Winograd-*n*-TD, für $n \leq 27$, in ihre Sub-TDs

n	**Sub-TD**																	
	GR	***1***	*NV-2*	***2***	*NV-3*	***3***	*NV-4*	***4***	*NV-5*	***5***	*NV-6*	***6***	*NV-7*	***7***	*NV-8*	***8***	*NV-9*	***9***
1	1																	
2	1	1																
3	1	3																
4	1	2	2															
5	1	2	2	1														
6	1	3		3														
7	1	5		1	2													
8	1	7			2	1												
9	1	9				3												
10	1	8	2			1	2											
11	1	7	4				2	1										
12	1	6	6					3										
13	1	6	6	1				1	2									
14	1	6	6	2					2	1								
15	1	6	6	3						3								
16	1	7	4	5						1	2							
17	1	8	2	7							2	1						
18	1	9		9								3						
19	1	11		7	2							1	2					
20	1	13		5	4								2	1				
21	1	15		3	6									3				
22	1	17		2	6	1								1	2			
23	1	19		1	6	2									2	1		
24	1	21			6	3										3		
25	1	23			4	5										1	2	
26	1	25			2	7											2	1
27	1	27				9												3

Analog zur Berechnung der Karatsuba-n-TD (siehe Kapitel 4.3.2 und 4.3.3) kann die GC der Winograd-n-TD mittels ***Algorithmus 3*** berechnet werden. In diesem Fall wird aber **Tabelle 26** als Input-Daten angegeben**.** Die Herleitung des XOR-Aufwandes der Berechnung der Winograd-n-GR, der für den ***Algorithmus 3*** notwendig ist, wird als eine selbstständige Aufgabe im nächsten Abschnitt gelöst.

4.4.2. Berechnung der Großen Raute der n-TD

Zuerst wurde die Darstellbarkeit der *n*-GR mittels der GR der kleineren Polynom-Längen geprüft. Die GR aller *n*-TD bis 81-Bits Polynome wurden analysiert, um die GR-Eigenschaften zu beleuchten. Die Ergebnisse der Analyse wurden auf weiteren *n*-TD, $81<n\leq 243$, geprüft.
Es wurde folgendes festgestellt:
Merkmal 1 der Karatsuba-n-TD (siehe **(114)**) und damit die Formeln **(90)-(93)** sind ohne Beschränkung gültig
Merkmal 2 der Karatsuba-n-TD ist auch hier gültig: Die linke n-GR-Seite kann mittels der r-GR, $r<n$, und der anderen, von den gefüllten Zellen gebildeten, geometrischen Figuren dargestellt werden. Die Anzahl der Figuren und deren geometrische Formen sind von der Binärdarstellung der Polynom-Länge *n* abhängig.

Abhängig von *n* wurden alle *n*-GR in 3 Gruppen aufgeteilt. **Tabelle 27** bietet eine kurze Beschreibung dieser Gruppen.

Tabelle 27: Winograd-n-GR-Gruppen

n-GR-Gruppe	*n*		Beispiele *n*
1	$n=\{3^i, 2\cdot 3^i\}$	$i\geq 0$	$n=\{3, 3^1, 3^2, 3^3, 3^4, ...\}$ $n=\{3, 2\cdot 3^1, 2\cdot 3^2, 2\cdot 3^3, 2\cdot 3^4, ...\}$
2	$n = 3^i + 2\cdot\sum_{j=1}^{i_{min}-1} 3^{i-j} + r$	$1 \leq r \leq 2\cdot 3^{i-i_{min}}$, $i \geq i_{min} \geq 1$, $\sum_{j=a}^{b<a} f(j) = 0$	$i_{min}=1 \Rightarrow\ i\geq 1,\ 1 \leq r \leq 2\cdot 3^{i-1}$, $n=3^i+r=\{4..5, 10..15, 28..45, 82..135, ...\}$ $i_{min}=2 \Rightarrow\ i\geq 2,\ 1 \leq r \leq 2\cdot 3^{i-2}$, $n=3^i+2\cdot 3^{i-1}+r=\{16..17, 46..51, 136..153, ...\}$ $i_{min}=3 \Rightarrow\ i\geq 3,\ 1 \leq r \leq 2\cdot 3^{i-3}$, $n=3^i+2\cdot 3^{i-1}+2\cdot 3^{i-2}+r=\{52..53, 154..159, ...\}$ $i_{min}=4 \Rightarrow\ i\geq 4,\ 1 \leq r \leq 2\cdot 3^{i-4}$, $n=3^i+2\cdot 3^{i-1}+2\cdot 3^{i-2}+2\cdot 3^{i-3}+r=\{160..161, ...\}$
3	$n = 2\cdot 3^i + 2\cdot\sum_{j=1}^{i_{min}-1} 3^{i-j} + r =$ $= 2\cdot\sum_{j=0}^{i_{min}-1} 3^{i-j} + r$		$i_{min}=1 \Rightarrow\ i\geq 1,\ 1 \leq r \leq 2\cdot 3^{i-1}$, $n=2\cdot 3^i+r=\{7..8, 19..24, 55..72, 163..216, ...\}$ $i_{min}=2 \Rightarrow\ i\geq 2,\ 1 \leq r \leq 2\cdot 3^{i-2}$, $n=2\cdot 3^i+2\cdot 3^{i-1}+r=\{25..26, 73..78, 217..234, ...\}$ $i_{min}=3 \Rightarrow\ i\geq 3,\ 1 \leq r \leq 2\cdot 3^{i-3}$, $n=2\cdot 3^i+2\cdot 3^{i-1}+2\cdot 3^{i-2}+r=\{79..80, 235..240, ...\}$ $i_{min}=4 \Rightarrow\ i\geq 4,\ 1 \leq r \leq 2\cdot 3^{i-4}$, $n=2\cdot 3^i+2\cdot 3^{i-1}+2\cdot 3^{i-2}+2\cdot 3^{i-3}+r=\{241..242, ...\}$

Jetzt werden alle *n-GR*-Gruppen einzeln analysiert.

***n*-GR-Gruppe 1**
Die Beschreibung dieser Gruppe ist zu der Beschreibung der Karatsuba-*n-GR* Gruppe 1 (siehe Kapitel 4.3.4.2) identisch: alle *n-GR*-Zeilen haben *n* ununterbrochen gefüllte Zellen, die zusammen eine geometrische Figur bilden, die einem Parallelogramm ähnlich ist. **Abbildung 20** stellt den typischen Vertreter dieser Gruppe dar. Formeln **(114)**-**(124)** sind hier ohne Beschränkung gültig.

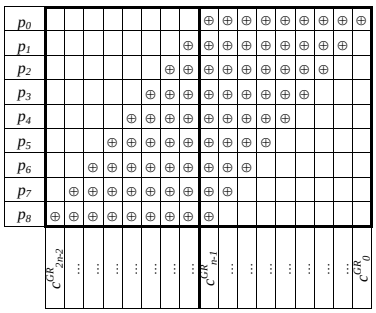


Abbildung 20: Vertreter der n-GR-Gruppe 1: 9-GR, $n= 3^2$

***n*-GR-Gruppe 2**

Eine *n*-GR gehört zur Gruppe 2, wenn *n* mit der Formel **(141)** beschrieben werden kann. **Abbildung 21** stellt einen typischen Vertreter dieser n-GR-Gruppe dar.

$$n = 3^i + 2 \cdot \sum_{j=1}^{i_{\min}-1} 3^{i-j} + r\,,\ 1 \le r \le 2 \cdot 3^{i-i_{\min}}\,,\quad i \ge i_{\min} = 1,\quad \text{wobei } \sum_{j=a}^{b<a} f(j) = 0 \qquad \textbf{(141)}$$

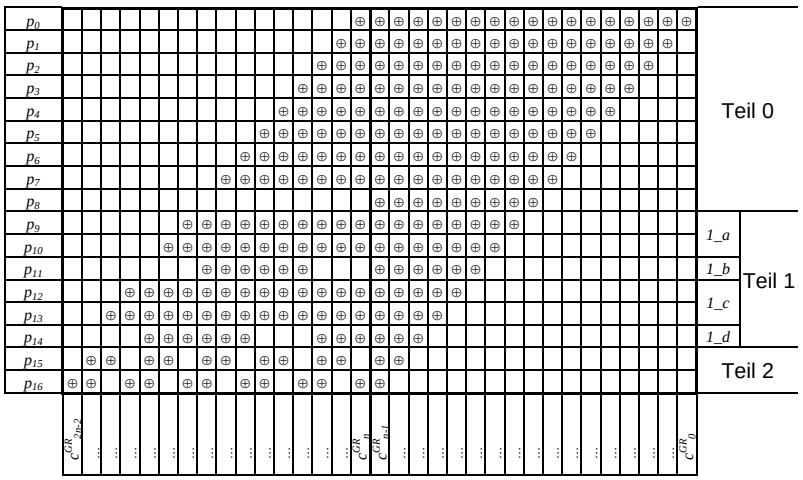


Abbildung 21: Vertreter der n-GR-Gruppe 2: 17-GR, $n=17=3^2+2\cdot3^1+2$, $i_{min}=i=r=2$

Die n-GR dieser Gruppe bestehen aus $(i_{min}+1)$ Teilen:

• **Teil 0** besteht aus 3^i Zeilen, wobei die $(n-3^i)$ davon $L=2\cdot3^i$ ununterbrochen gefüllte Zellen haben. Die restlichen $(2\cdot3^i-n)$ Zeilen haben $L=3^i$.

• Jeder **Teil j** (*1≤j≤i_min-1*) besteht aus $2 \cdot 3^{i-j}$ Zeilen, und kann folgendermaßen weiter aufgeteilt werden:

- **Teil j_a** besteht aus $N_{j\_a} = n - 3^i - 2 \cdot \sum_{k=1}^{j} 3^{i-k}$ Zeilen, die $L_{j\_a}=2 \cdot 3^j$ ununterbrochen gefüllte Zellen haben
- **Teil j_b** besteht aus $N_{j\_b}=2 \cdot 3^j - n$ Zeilen, in denen $K_{j\_b}=2 \cdot 3^{j-1}$ gleiche Parallelogramme platziert sind. Die Grundseite jedes Parallelogramms besteht aus $L_{j\_b}=2 \cdot 3^{i-j}$ ununterbrochen gefüllten Zellen.
- **Teil j_c** ist zu dem **Teil j_a** identisch
- **Teil j_d** ist zu dem **Teil j_b** identisch

• **Teil *i_min*** besteht aus *r* Zeilen, die $K_{i_{\min}} = 2 \cdot 3^{i_{\min}-1}$ *r-GR*-Figuren beinhalten.

n-GR-Gruppe 3

Eine *n*-GR gehört zu der Gruppe 3, wenn *n* mit der Formel **(142)** beschrieben werden kann. **Abbildung 22** stellt einen typischen Vertreter dieser n-GR-Gruppe dar.

$$n = 2 \cdot \sum_{j=0}^{i_{\min}-1} 3^{i-j} + r \text{ ,} \qquad 1 \le r \le 2 \cdot 3^{i-i_{\min}},\ i \ge i_{\min} = 1, \qquad \sum_{j=a}^{b<a} f(j) = 0$$

(142)

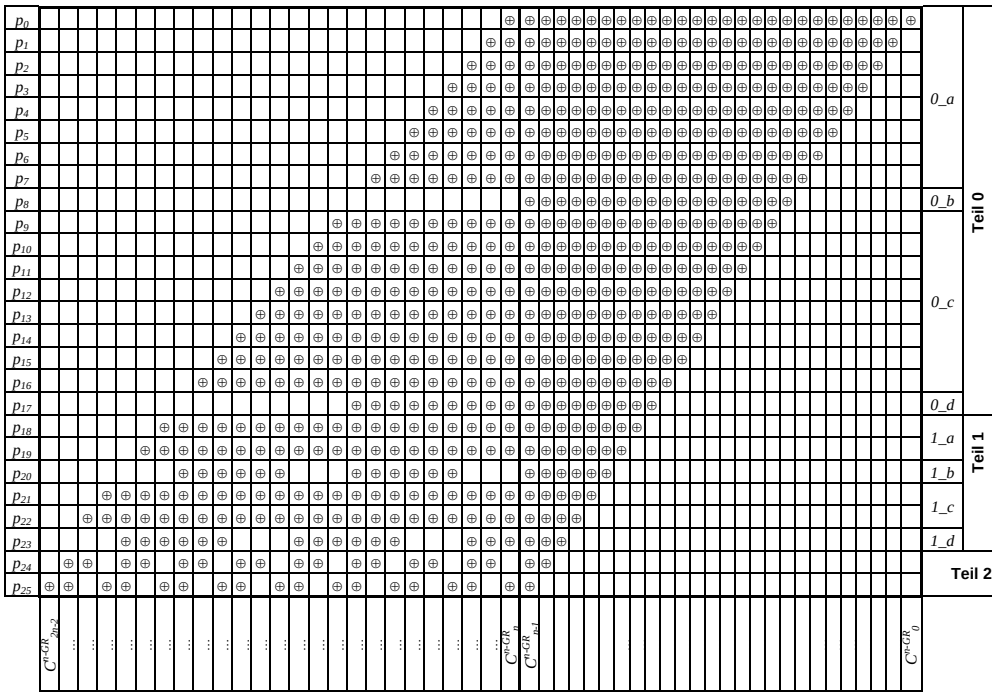


Abbildung 22: Vertreter der n-GR-Gruppe 3: 26-GR, $n=26=2 \cdot 3^2+2 \cdot 3^1+2$, $i_{min}=i=r=2$

Die n-GR dieser Gruppe bestehen aus *(i_{min} +1)* Teilen:

• Jeder **Teil *j*** (*$0 \le j \le i_{min}-1$*) besteht aus *$2 \cdot 3^{i-j}$* Zeilen, und kann folgendermaßen weiter aufgeteilt werden:

- **Teil *j_a*** besteht aus $N_{j\_a} = 3^{i-j} - 3^{i-(i_{\min}-1)} + r$ Zeilen mit $L_{j\_a}=3^{i+1}$
- **Teil *j_b*** besteht aus $N_{j\_b} = 3^{i-(i_{\min}-1)} - r$ Zeilen, in denen $K_{j\_b} = 3^j$ gleiche Parallelogramme platziert sind. Die Grundseite jedes Parallelogramms besteht aus $L_{j\_b} = 2 \cdot 3^{i-j}$ ununterbrochen gefüllten Zellen.
- **Teil *j_c*** ist zu dem **Teil *j_a*** identisch
- **Teil *j_d*** ist zu dem **Teil *j_b*** identisch

• **Teil i_{min}** besteht aus *r* Zeilen, die $K_{i_{\min}} = 3^{i_{\min}}$ *r-GR*-Figuren[16] beinhalten.

Nach dem Vergleich der n-GR-Gruppen 2 und 3 wurden die folgenden gemeinsamen Merkmale festgestellt:

- die Beschreibung aller **Teile *j***, *$1 \le j \le i_{min}$*, bei beiden n-GR-Gruppen ist bis zu den Werten der Parameter N_j, L_j und K_j identisch
- Für die Teile ***j_a*** and ***j_c*** , *$1 \le j \le i_{min}-1$* , gilt folgendes: jede Spalte (ab c_{2n-3-r}) beinhaltet in sich alle TP ihrer linken Nachbar-Spalte und kann ein weiteres Teil-Produkt beinhalten. Die *iterative* (evtl. *voll iterative*) Berechnung ist hier optimal (siehe Kapitel 3.9). Zu den anderen n-GR-Teilen außer linker Seite *r-GR*-Figur Nr.1 sind jetzt die Kriterien (***K***) verwendbar.
- Die Anzahl der ununterbrochen gefüllten Zellen in jedem **Teil *j_b*** and ***j_d*** , *$1 \le j \le i_{min}-1$* , kann nicht kleiner als 6 sein:
$$\min\left(L_{j\_b|\ 1\le j\le i_{\min}-1} = 2 \cdot 3^{i-j}\right) = 2 \cdot 3^{\min(i)-\max(j)} = 2 \cdot 3^{i_{\min}-(i_{\min}-1)} = 6$$
Deswegen gehört jeder Teil ***j_b*** und ***j_d***, *$1 \le j \le i_{min}-1$*, nach ***Kriterien (K)*** zu dem *iterativen* TD-Teil
- Der ganze **Teil 0** bei beiden n-GR-Gruppen gehört nach ***Kriterien (K)*** dem *iterativen* Sub-Teil der TD
- Die Position der Parallelogramme[17] aus der Teilen ***j_b*** und ***j_d*** begünstigt die *iterative* Berechnung zusätzlich: im Teil ***j_b*** tragen die Parallelogramme Nr. 1 bis 3 (hier auch nummeriert von links nach rechts) mit ihren beiden Seiten zu dem gesamten XOR-Aufwand bei und die restlichen (K_{j_b}-3) Parallelogramme nur mit ihren rechten Seiten; im Teil *j_d* tragen die Parallelogramme Nr. 1 bis 2 mit ihren

[16] Auch hier werden alle r-GR-Figuren von links nach rechts nummeriert: r-GR-Figur Nr.1, …, *r*-GR-Figur Nr. $K_{i_{\min}}$

[17] Ähnlich wie bei der Karatsuba-*n*-TD (siehe Kapitel 4.3.4) sind die linken Seiten der Parallelogramme unter den Seiten der oben stehenden Parallelogramme platziert.

beiden Seiten zu dem gesamten XOR-Aufwand bei und die restlichen (K_{j_d}-2) Parallelogramme nur mit ihren rechten Seiten;

- Der einzige Inhalt der linken *r* Spalten der n-GR ist die linke Seite der *r-GR*-Figur Nr.1, was die *separate* Berechnung des Teils i_{min} begünstigt; die Position der anderen *r-GR*-Figuren, die sich unter den Parallelogramm-Seiten des Teils (i_{min}-*1*) befinden, begünstigt die *iterative* Berechnung des Teils i_{min}. Daraus folgt, dass die Anordnung der *r-GR*-Figur Nr.1 zum *iterativen* oder *separaten* TD-Teil als ein Sonderfall analysiert werden muss.

Aus den oben genannten Merkmalen folgt, dass der XOR-Aufwand der optimalen Berechnung der linken n-GR-Seite für beide n-GR-Gruppen folgendermaßen beschrieben werden kann:

$$\#XOR^{optimal}_{c^{n-GR}_{2n-2}..c^{n-GR}_{n}} = \#XOR^{iterativ}_{\text{Teil }0} + \begin{cases} 0, i_{\min}=1 \\ \sum_{j=1}^{i_{\min}-1}\left(\#XOR^{iterativ}_{j\_a} + \#XOR^{iterativ}_{j\_b} + \#XOR^{iterativ}_{j\_c} + \#XOR^{iterativ}_{j\_d} \right), i_{\min}>1 \end{cases} + \#XOR^{optimal}_{\text{Teil } i_{\min}} =$$

$$= \#XOR^{iterativ}_{\text{Teil }0} + \#XOR^{optimal}_{\text{Teil } i_{\min}} + \begin{cases} 0, \ i_{\min}=1 \\ \left(\underbrace{\#XOR^{iterativ}_{1\_a}}_{=N_{1\_a}} + \underbrace{\#XOR^{iterativ}_{1\_b}}_{=(K_{1\_b}+2)N_{1\_b}-1+\begin{cases} -N_{1\_b}, & n-\text{GR aus Gruppe2} \\ 0, & n-\text{GR aus Gruppe3} \end{cases}} + \right. \\ \left. + \underbrace{\#XOR^{iterativ}_{1\_c}}_{=N_{1\_a}} + \underbrace{\#XOR^{iterativ}_{1\_d}}_{=(K_{1\_b}+1)N_{1\_b}} \right), \ i_{\min}>1 \end{cases} +$$

$$+ \begin{cases} 0, \ i_{\min}=1 \\ \sum_{j=2}^{i_{\min}-1}\left(\underbrace{\#XOR^{iterativ}_{j\_a}}_{=N_{j\_a}} + \underbrace{\#XOR^{iterativ}_{j\_b}}_{=(K_{j\_b}+2)N_{j\_b}} + \underbrace{\#XOR^{iterativ}_{j\_c}}_{=N_{j\_a}} + \underbrace{\#XOR^{iterativ}_{j\_d}}_{=(K_{j\_b}+1)N_{j\_b}} \right), \ i_{\min}>2 \end{cases} =$$

$$= \#XOR^{iterativ}_{\text{Teil }0} + \#XOR^{optimal}_{\text{Teil } i_{\min}} +$$

$$+ \begin{cases} 0, \ i_{\min}=1 \\ 2\cdot N_{1\_a} + (2\cdot K_{1\_b}+3)\cdot N_{1\_b} - 1 + \begin{cases} -N_{1\_b}, & i_{\min}>1, \ n-\text{GR aus Gruppe2} + \\ 0, & i_{\min}>1, \ n-\text{GR aus Gruppe3} \end{cases} \end{cases}$$

$$+ \begin{cases} 0, \ i_{\min}=1 \\ \sum_{j=2}^{i_{\min}-1}\left(2\cdot N_{j\_a} + (2\cdot K_{j\_b}+3)\cdot N_{j\_b}\right), \ i_{\min}>2 \end{cases}$$

(143)

Der **Teil 0** trägt folgende Anzahl an XOR-Operationen zu dem gesamten XOR-Aufwand bei:

$$\#XOR_{\text{Teil }0}^{iterativ}=\begin{cases}\underbrace{\left(n-3^i\right)}_{\text{Teil }0\_a}+\underbrace{\left(2\cdot 3^i-n-1\right)}_{\text{Teil }0\_b}=3^i-1,\ \ n-\text{GR aus Gruppe 2}\\ \underbrace{2\cdot\left(3^{i-j}-3^{i-(i_{\min}-1)}+r\right)}_{\text{Teil }0\_a\text{ und}0\_c}+\underbrace{\left(3^{i-(i_{\min}-1)}-r\right)\cdot 2-1}_{\text{Teil }0\_b\text{ und}0\_d}=\\ \qquad\qquad =2\cdot 3^i-1,\ \ n-\text{GR aus Gruppe 3}\end{cases}$$

(144)

Aus **(143)**, **(144)** und den entsprechenden Parameter-Werten N_j, K_j ergibt sich folgender XOR-Aufwand für beiden n-GR-Gruppen:

$$\#XOR_{c_{2n-2}^{GR}\ldots c_n^{GR}}^{optimal}=\#XOR_{\text{Teil }i_{\min}}^{optimal}+\begin{cases}\begin{cases}3^i-1,\ n-\text{GR aus Gruppe 2},\ i_{\min}=1\\ 2\cdot 3^i-1,\ n-\text{GR aus Gruppe 3},\ i_{\min}=1\end{cases}\\ 4\cdot 3^i+3^{i-i_{\min}+1}\cdot\left(i_{\min}-5\right)-r\left(2\cdot 3^{i_{\min}-1}+i_{\min}-4\right)-2+\\ \quad+\begin{cases}0,\ n-\text{GR aus Gruppe 2},\ i_{\min}\geq 2\\ 2\cdot 3^i-r\cdot 3^{i_{\min}-1},\ n-\text{GR aus Gruppe 3},\ i_{\min}\geq 2\end{cases}\end{cases}$$

(145)

Die ausführliche Herleitung der Formel **(145)** ist im Anhang 2 gegeben.

Die Ermittlung des Wertes $\#XOR_{\text{Teil }i_{\min}}^{optimal}$ in **(145)** ist eine aufwendige Aufgabe, die den Vergleich mehrerer Ablaufpläne benötigt. Wenn dieser Wert durch seinen minimalen Wert ersetzt wird und die resultierende Chip-Fläche des Polynom-Multiplizierers nicht die kleinste (im Vergleich zu den anderen MM) ist, hat die Kenntnis dieses Wertes keine große praktische Bedeutung. In dieser Arbeit wurde die untere Grenze des XOR-Aufwandes der Berechnung des Teils i_{min} folgendermaßen eingeführt:

$$\#XOR_{\text{Teil }i_{\min}}^{optimal}\geq 0$$

(146)

Daraus folgt:

$$\#XOR_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}}^{optimal}\geq\ \#XOR_{c_{2n-2}^{\text{Teil }0\ldots(i_{\min}-1)}\ldots c_n^{\text{Teil }0\ldots(i_{\min}-1)}}^{voll\ iterativ}=$$

$$=\begin{cases}\begin{cases}3^i-2,\ n-\text{GR aus Gruppe 2},\ i_{\min}=1\\ 2\cdot 3^i-2,\ n-\text{GR aus Gruppe 3},\ i_{\min}=1\end{cases}\\ 4\cdot 3^i+3^{i-i_{\min}+1}\cdot\left(i_{\min}-5\right)-r\left(2\cdot 3^{i_{\min}-1}+i_{\min}-4\right)-\\ \quad-3+\begin{cases}0,\ n-\text{GR aus Gruppe 2},\ i_{\min}\geq 2\\ 2\cdot 3^i-r\cdot 3^{i_{\min}-1},\ n-\text{GR aus Gruppe 3},\ i_{\min}\geq 2\end{cases}\end{cases}$$

(147)

Formel **(148)** zeigt die untere Grenze des XOR-Aufwandes der optimalen Berechnung der ganzen GR aller drei *n*-GR-Gruppen:

$$
\#\,XOR^{optimal}_{n-GR} \geq \begin{cases} \#\,XOR^{volliterativ}_{n-GR} = 2n-3,\ n-\text{GR aus Gruppe 1} \\ n-1+\#\,XOR^{voll\ iterativ}_{c^{\text{Teil }0..(i_{\min}-1)}_{2n-2}\ldots c^{\text{Teil }0..(i_{\min}-1)}_{n}} = \\ \quad = n-1+\begin{cases} 3^i-2,\ n-\text{GR aus Gruppe2},\ i_{\min}=1 \\ 4\cdot 3^i+3^{i-i_{\min}+1}\cdot(i_{\min}-5)- \\ \quad -r\cdot\left(2\cdot 3^{i_{\min}-1}+i_{\min}-4\right)-3,\ n-\text{GR aus Gruppe2},\ i_{\min}\geq 2 \end{cases} \\ n-1+\#\,XOR^{voll\ iterativ}_{c^{\text{Teil }0..(i_{\min}-1)}_{2n-2}\ldots c^{\text{Teil }0..(i_{\min}-1)}_{n}} = \\ \quad = n-1+\begin{cases} 2\cdot 3^i-2, n-\text{GR aus Gruppe 3},\ i_{\min}=1 \\ 2\cdot 3^{i+1}+3^{i-i_{\min}+1}\cdot(i_{\min}-5)- \\ \quad -r\cdot\left(3^{i_{\min}}+i_{\min}-4\right)-3, n-\text{GR aus Gruppe 3},\ i_{\min}\geq 2 \end{cases} \end{cases}
$$

(148)

Auf ähnliche Weise kann die untere Grenze der optimalen Berechnung der *nicht vollen* GR abgeschätzt werden:

$$
\#\,XOR^{optimal}_{\text{Teil } i_{\min},NV-n-GR} \geq 0 \Rightarrow
$$

$$
\#\,XOR^{optimal}_{NV-n-GR} \geq \begin{cases} \#\,XOR^{volliterativ}_{NV-n-GR} = 2n-5,\ n-\text{GR aus Gruppe 1} \\ n-2+\#\,XOR^{voll\ iterativ}_{c^{\text{Teil }0..(i_{\min}-1)}_{2n-2}\ldots c^{\text{Teil }0..(i_{\min}-1)}_{n}},\ n-\text{GR aus Gruppe 2} \\ n-2+\#\,XOR^{voll\ iterativ}_{c^{\text{Teil }0..(i_{\min}-1)}_{2n-2}\ldots c^{\text{Teil }0..(i_{\min}-1)}_{n}},\ n-\text{GR aus Gruppe 3} \end{cases}
$$

(149)

Die untere Grenze der Berechnung des Spalten-Wertes $c^{n\text{-}GR}{}_{n\text{-}1}$ vom Wert $c^{n\text{-}GR}{}_{n}$ (auch des Spalten-Wertes $c^{NV\text{-}n\text{-}GR}{}_{n\text{-}1}$ vom Wert $c^{NV\text{-}n\text{-}GR}{}_{n}$) nach dem optimalen Ablaufplan kann folgendermaßen abgeschätzt werden:

$$
\begin{aligned} &\#\,XOR^{optimal}_{c^{n-GR}_{n-1}} \geq 1, && \#\,XOR^{voll\ iterativ}_{c^{n-GR}_{n-1}} \geq 1 \\ &\#\,XOR^{optimal}_{c^{NV-n-GR}_{n-1}} \geq 1, && \#\,XOR^{voll\ iterativ}_{c^{NV-n-GR}_{n-1}} \geq 1 \end{aligned}
$$

(150)

Die Formeln **(147) – (150)** wurden im ***Algorithmus 2*** benutzt, um die untere Grenze der Chip-Fläche der iterativ optimierten Winograd-MM zu ermitteln. Nach der Analyse der von ***Algorithmus 2*** gelieferten Ergebnisse, wo mehrere MM

verglichen wurden, wurde es festgestellt, dass die iterativ optimierte Winograd-MM für die Polynom-Länge *n≤600*, mit n aus n-GR-Gruppen 2 und 3, nie als Implementierungs-Kandidat bestimmt war. Aus diesem Grund hat die Ermittlung der Formeln des XOR-Aufwandes mit exaktem Wert $\#^{optimal}XOR_{\text{Teil } i_{\min}}$ keine praktische Bedeutung und wurde in Rahmen dieser Arbeit nicht weiter geführt.

4.4.3. Iterative Optimierung der Winograd-*n*-Segment-TD

Analog zu den Karatsuba-*n*-Segment-TD wurden im Rahmen dieser Arbeit die GK der *iterativ optimierten* Berechnung der Winograd-*n*-Segment-TD, *2≤n≤16*, ermittelt. **Tabelle 28** zeigt die GK für alle untersuchten und im ***Algorithmus 2*** benutzten Segmentierungen. Nach der Analyse der Ergebnisse des ***Algorithmus 2*** wurde festgestellt, dass nur die Aufteilung der Polynome in 3 und 6 Segmente eine praktische Bedeutung für IHP-Technologien hat.

Tabelle 28: Gatter-Komplexität Winograd-*n*-Segment-TD, mit *2≤n≤16*

n	*Gatter-Komplexität*	*delay,T_{XOR}*
2	$GC_{2m}=3\cdot GC_m+(7m-3)_{XOR}$	3
3	$GC_{3m}=6\cdot GC_m+(18m-6)_{XOR}$	4
4	$GC_{4m}=10\cdot GC_m+(37m-12)_{XOR}$	5
5	$GC_{5m}=14\cdot GC_m+(56m-18)_{XOR}$	6
6	$GC_{6m}=18\cdot GC_m+(72m-19)_{XOR}$	7
7	$GC_{7m}=25\cdot GC_m+(111m-35)_{XOR}$	8
8	$GC_{8m}=31\cdot GC_m+(137m-38)_{XOR}$	9
9	$GC_{9m}=36\cdot GC_m+(156m-38)_{XOR}$	10
10	$GC_{10m}=46\cdot GC_m+(213m-55)_{XOR}$	11
11	$GC_{11m}=54\cdot GC_m+(254m-63)_{XOR}$	12
12	$GC_{12m}=60\cdot GC_m+(285m-67)_{XOR}$	13
13	$GC_{13m}=70\cdot GC_m+(338m-80)_{XOR}$	14
14	$GC_{14m}=78\cdot GC_m+(381m-91)_{XOR}$	15
15	$GC_{15m}=84\cdot GC_m+(415m-94)_{XOR}$	16
16	$GC_{16m}=94\cdot GC_m+(477m-114)_{XOR}$	17

4.5. Optimierung der Karatsuba- und Winograd-MM für *n*-Term-Polynome mit Ersatz der Sub-TD

Eine weitere Möglichkeit, die Fläche und/oder den Energieverbrauch des Polynom-Multiplizierers zu reduzieren, kann auf Basis der Karatsuba-oder der Winograd n-Term Multiplikation vorgenommen werden.
Die Hauptidee ist, dass alle Sub-Tabellen, aus denen laut **Tabelle 15** eine Karatsuba-n-TD besteht (bzw. **Tabelle 26** für Winograd-n-TD), durch die TD einer

anderen – optimalen MM – ersetzt werden können. Es wird am Beispiel der Karatsuba-4-TD erklärt.

Tabelle 29 zeigt die Aufteilung der Karatsuba-4-TD in Sub-TD laut **Tabelle 15**: die Karatsuba-4-TD besteht aus Großer Raute, zwei 1-TD und einer 2-TD.

Tabelle 29: Aufteilung der Karatsuba-4-TD in Sub-TD entsprechend **Tabelle 15**

$a_0 \cdot b_0$				⊕	⊕	⊕	⊕	Große Raute
$a_1 \cdot b_1$			⊕	⊕	⊕	⊕		
$a_2 \cdot b_2$		⊕	⊕	⊕	⊕			
$a_3 \cdot b_3$	⊕	⊕	⊕	⊕				
$(a_0 \oplus a_2) \cdot (b_0 \oplus b_2)$				⊕	⊕			1-TD
$(a_1 \oplus a_3) \cdot (b_1 \oplus b_3)$			⊕	⊕				1-TD
$(a_0 \oplus a_1) \cdot (b_0 \oplus b_1) = \alpha_0 \cdot \beta_0$				⊕		⊕		2-TD
$(a_2 \oplus a_3) \cdot (b_2 \oplus b_3) = \alpha_1 \cdot \beta_1$		⊕		⊕				
$(a_0 \oplus a_1 \oplus a_2 \oplus a_3) \cdot (b_0 \oplus b_1 \oplus b_2 \oplus b_3) = (\alpha_0 \oplus \alpha_1) \cdot (\beta_0 \oplus \beta_1)$				⊕				
	c_6	c_5	c_4	c_3	c_2	c_1	c_0	

Die Sub-Tabelle Karatsuba-2-TD in **Tabelle 29** kann durch die 2-TD, die einer anderer MM entspricht, z.B. klassischer MM, folgendermaßen ersetzt werden:

$\alpha_0 \cdot \beta_0$		⊕	⊕
$\alpha_1 \cdot \beta_1$	⊕	⊕	
$(\alpha_0 \oplus \alpha_1) \cdot (\beta_0 \oplus \beta_1)$		⊕	

⟺

$\alpha_0 \cdot \beta_0$			⊕
$\alpha_0 \cdot \beta_1$		⊕	
$\alpha_1 \cdot \beta_0$		⊕	
$\alpha_1 \cdot \beta_1$	⊕		

Tabelle 30 zeigt die Karatsuba-4-TD mit der 2-TD nach klassischer MM.

Tabelle 30: Karatsuba-4-TD mit klas-2-TD

$a_0 \cdot b_0$				⊕	⊕	⊕	⊕	Große Raute
$a_1 \cdot b_1$			⊕	⊕	⊕	⊕		
$a_2 \cdot b_2$		⊕	⊕	⊕	⊕			
$a_3 \cdot b_3$	⊕	⊕	⊕	⊕				
$(a_0 \oplus a_2) \cdot (b_0 \oplus b_2)$				⊕	⊕			1-TD
$(a_1 \oplus a_3) \cdot (b_1 \oplus b_3)$			⊕	⊕				1-TD
$\alpha_0 \cdot \beta_0 = (a_0 \oplus a_1) \cdot (b_0 \oplus b_1)$						⊕		2-TD
$\alpha_0 \cdot \beta_1 = (a_0 \oplus a_1) \cdot (b_2 \oplus b_3)$				⊕				
$\alpha_1 \cdot \beta_0 = (a_2 \oplus a_3) \cdot (b_0 \oplus b_1)$				⊕				
$\alpha_1 \cdot \beta_1 = (a_2 \oplus a_3) \cdot (b_2 \oplus b_3)$		⊕						
	c_6	c_5	c_4	c_3	c_2	c_1	c_0	

Auch eine nicht vollständige *NV-i*-TD, *i<n*, kann durch eine äquivalente Sub-TD, die einer anderen MM entspricht, ersetzt werden. Dafür muss noch eine zusätzliche Zeile mit dem Inhalt der Zeile p_{i-1} der *i*-TD zugefügt werden. Dies ist in **Abbildung 23** am Beispiel Karatsuba-*NV-2*-TD schrittweise gezeigt. Nicht vollständige Karatsuba-*2*-TD wird durch eine äquivalente Sub-TD ersetzt. Diese

äquivalente Sub-TD wurde aus der klassischen MM mit dem Zufügen der Zeile p_1 der Karatsuba-2-TD ermittelt.

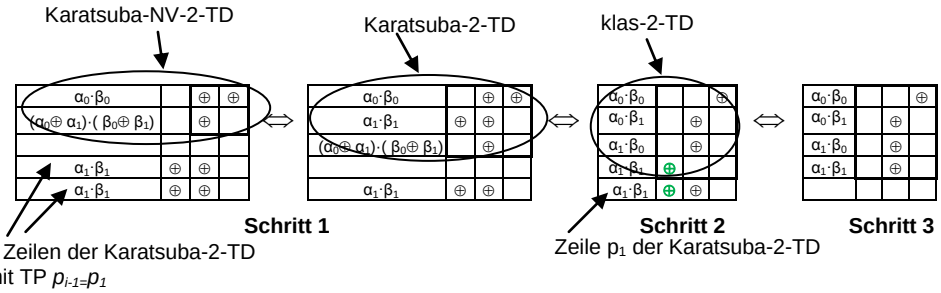


Abbildung 23: Karatsuba-NV-2-TD kann durch eine äquivalente TD (siehe Schritt 3) ersetzt werden. Diese äquivalente TD besteht aus der klass-2-TD und der Zeile p_1 der Karatsuba-2-TD.

Das Ersetzen der Sub-TD in Karatsuba- und Winograd-n-TD kann programmiert werden. Die Ermittlung der Gatter-Komplexität auf dieser Weise optimierten Multiplikationen wurden mittels ***Algorithmus 4*** berechnet.

Algorithmus 4

Input:
Polynom-Länge *n,* **Tabelle 15**,
GC für Karatsuba-i-TD, Karatsuba-NV-i-TD (bzw. Winograd-MM), optim-i-TD, *i*<*n*
Output: GC für Karatsuba-n-TD (bzw. Winograd-n-TD) mit
optimal ersetzten Sub-TD
Berechnung
for *each 2≤ i<n*

if **(**(*n-TD* beinhaltet k-fach die *i-TD*) **and**
(optimization_Param(GC(Karatsuba-i-TD)) >
optimization_Param(GC(optim-i-TD)))
• GC(Karatsuba-n-TD)=
=GC(Karatsuba-n-TD)+k·(-GC(Karatsuba-i-D)+GC(optim-i-TD))
end if

if **(**(*n-TD* beinhaltet k-fach die *NV-i-TD*) **and**
(optimization_Param(GC(Karatsuba-NV-i-TD)) >
optimization_Param(GC(optim-i-TD))+
+(Number_of_filled_cells_in_Row_p_{i-1}-1)$_{\text{XOR-Gatter}}$)
• GC(Karatsuba-n-TD)=
= GC(Karatsuba-n-TD)+k·(-GC(Karatsuba-NV-i-TD)+
+GC(optim-i-TD)+(Number_of_filled_cells_in_Row_p_{i-1}-1)$_{\text{XOR-Gatter}}$)
end if

end for

Die mit dem ***Algorithmus 4*** ermittelten GC für Karatsuba- und Winograd-n-TD für Polynom-Länge n≤600 wurden als Tabellen gespeichert und im ***Algorithmus 2*** benutzt. Es wurde festgestellt, dass diese beiden MM – *n*-Bits-Karatsuba-MM mit Ersatz der Sub-TD und *n*-Bits-Winograd-MM mit Ersatz der Sub-TD – keine praktische Bedeutung für die ECC-relevanten Polynom-Längen haben.
Die beschriebene Optimierung der Karatsuba- und Winograd-n-Segment-TD wurde in Rahmen dieser Arbeit nicht durchgeführt.

4.6. Iterative Optimierung der Montgomery-MM für 5-Term-Polynome

In diesem Kapitel wird die Montgomery-MF für 5-Term-Polynome [19] für die 5-Bits- und 5-Segment-Polynome (aus $GF(2^n)$) angewendet und iterativ optimiert.

Montgomery-MF für die Polynome von Grad 4 [19] ist folgende:

$$
\begin{aligned}
&a_4a_3a_2a_1a_0 \cdot b_4b_3b_2b_1b_0 = \\
&= \left(a_0 + a_1 \cdot X^1 + a_2 \cdot X^2 + a_3 \cdot X^3 + a_4 \cdot X^4\right)\left(b_0 + b_1 \cdot X^1 + b_2 \cdot X^2 + b_3 \cdot X^3 + b_4 \cdot X^4\right) = \\
&\quad = \left(a_0 + a_1 + a_2 + a_3 + a_4\right)\left(b_0 + b_1 + b_2 + b_3 + b_4\right)\left(X^5 - X^4 + X^3\right) + \\
&\quad + \left(a_0 - a_2 - a_3 - a_4\right)\left(b_0 - b_2 - b_3 - b_4\right)\left(X^6 - 2X^5 + 2X^4 - X^3\right) + \\
&\quad + \left(a_0 + a_1 + a_2 - a_4\right)\left(b_0 + b_1 + b_2 - b_4\right)\left(-X^5 + 2X^4 - 2X^3 + X^2\right) + \\
&\quad + \left(a_0 + a_1 - a_3 - a_4\right)\left(b_0 + b_1 - b_3 - b_4\right)\left(X^5 - 2X^4 + X^3\right) + \\
&\quad + \left(a_0 - a_2 - a_3\right)\left(b_0 - b_2 - b_3\right)\left(-X^6 + 2X^5 - X^4\right) + \\
&\quad + \left(a_0 + a_2 - a_4\right)\left(b_0 + b_2 - b_4\right)\left(-X^4 + 2X^3 - X^2\right) + \\
&\quad + \left(a_3 + a_4\right)\left(b_3 + b_4\right)\left(X^7 - X^6 + X^4 - X^3\right) + \\
&\quad + \left(a_0 + a_1\right)\left(b_0 + b_1\right)\left(-X^5 + X^4 - X^2 + X\right) + \\
&\quad + \left(a_0 - a_4\right)\left(b_0 - b_4\right)\left(-X^6 + 3X^5 - 4X^4 + 3X^3 - X^2\right) + \\
&\quad + a_4b_4\left(X^8 - X^7 + X^6 - 2X^5 + 3X^4 - 3X^3 + X^2\right) + \\
&\quad + a_3b_3\left(-X^7 + 2X^6 - 2X^5 + X^4\right) + \\
&\quad + a_1b_1\left(X^4 - 2X^3 + 2X^2 - X\right) + \\
&\quad + a_0b_0\left(X^6 - 3X^5 + 3X^4 - 2X^3 + X^2 - X + 1\right)
\end{aligned}
\qquad \textbf{(151)}
$$

Die Komplexität dieser Multiplikation wurde in [19] mit 13 Teil-Multiplikationen beschrieben. Die Anzahl der Additionen – 22 – wurde nur für die Berechnung der Operanden für die Teil-Multiplikation nach der Reihenfolge **(152)** angegeben:

$$
\begin{array}{ll}
a_0 + a_1 & b_0 + b_1 \\
a_0 - a_4 & b_0 - b_4 \\
a_3 + a_4 & b_3 + b_4 \\
(a_0 + a_1) - (a_3 + a_4) & (b_0 + b_1) - (b_3 + b_4) \\
(a_0 + a_1) + a_2 & (b_0 + b_1) + b_2 \\
a_2 + (a_3 + a_4) & b_2 + (b_3 + b_4) \\
(a_0 + a_1) + (a_2 + a_3 + a_4) & (b_0 + b_1) + (b_2 + b_3 + b_4) \\
a_0 - (a_2 + a_3 + a_4) & b_0 - (b_2 + b_3 + b_4) \\
(a_0 + a_1 + a_2) - a_4 & (b_0 + b_1 + b_2) - b_4 \\
(a_0 - a_2 - a_3 - a_4) + a_4 & (b_0 - b_2 - b_3 - b_4) + b_4 \\
(a_0 + a_1 + a_2 - a_4) - a_0 & (b_0 + b_1 + b_2 - b_4) - b_0
\end{array}
$$

(152)

Die optimierte Berechnung der Produkt-Segmente als eine Reihenfolge der Operationen ist in [19] nicht angegeben, es wird nur erwähnt, dass verschiedene Umwandlungen der Formel **(151)** die Anzahl der Additionen reduzieren können. Auch wurde in [19] erwähnt, dass die Anwendung der Formel **(151)** für die Multiplikation der Polynome aus *GF(2^n)* eine kleinere Anzahl an Additionen benötigt.

Formel **(153)** stellt die Gatter-Komplexität der Montgomery-MM (siehe **(151)**) mit der Berücksichtigung der in [19] angegebener Reihenfolge **(152)**:

$$
\begin{aligned}
GC_{5m}^{\substack{\text{5-Term}\\ \text{Montgomery}\\ \text{MM}}} &= 13 \cdot GC_m^{\substack{\text{MM des}\\ m\text{-TM}}} + \left(\underbrace{22m}_{\text{TM-Eingänge}} + \underbrace{32(2m-1)}_{\substack{\text{Berechnung aller}\\ \text{TP-Summen}}} + \underbrace{8(m-1)}_{\substack{\text{Berechnung}\\ \text{Produkt-Segmente}}} \right)_{XOR} = \\
&= 13 \cdot GC_m^{\substack{\text{MM des}\\ m\text{-TM}}} + (94m - 37)_{XOR}
\end{aligned}
$$

(153)

$$
delay_{5m}^{\text{Montgom}} = delay_m + 6 \cdot T_{XOR}
$$

(154)

Jetzt wird die Formel **(151)** für die Multiplikation der 5-Bits-Polynome tabellarisch dargestellt (siehe **Tabelle 31**), um die Montgomery-MM iterativ zu optimieren.

Tabelle 31: Montgomery-5-TD

$a_0 \cdot b_0 = p_0$			⊕	⊕	⊕		⊕	⊕	⊕
$(a_0 \oplus a_1) \cdot (b_0 \oplus b_1) = p_{01}$				⊕	⊕		⊕	⊕	
$(a_0 \oplus a_4) \cdot (b_0 \oplus b_4) = p_{04}$			⊕	⊕		⊕	⊕		
$a_4 \cdot b_4 = p_4$	⊕	⊕	⊕		⊕	⊕	⊕		
$(a_3 \oplus a_4) \cdot (b_3 \oplus b_4) = p_{34}$		⊕	⊕		⊕	⊕			
$(a_0 \oplus a_1 \oplus a_2 \oplus a_3 \oplus a_4) \cdot (b_0 \oplus b_1 \oplus b_2 \oplus b_3 \oplus b_4) = p_{01234}$				⊕	⊕	⊕			
$(a_0 \oplus a_2 \oplus a_3 \oplus a_4) \cdot (b_0 \oplus b_2 \oplus b_3 \oplus b_4) = p_{0234}$			⊕			⊕			
$(a_0 \oplus a_1 \oplus a_3 \oplus a_4) \cdot (b_0 \oplus b_1 \oplus b_3 \oplus b_4) = p_{0134}$				⊕		⊕			
$(a_0 \oplus a_1 \oplus a_2 \oplus a_4) \cdot (b_0 \oplus b_1 \oplus b_2 \oplus b_4) = p_{0124}$				⊕			⊕		
$(a_1 \oplus a_2 \oplus a_4) \cdot (b_1 \oplus b_2 \oplus b_4) = p_{124}$					⊕		⊕		
$(a_0 \oplus a_2 \oplus a_3) \cdot (b_0 \oplus b_2 \oplus b_3) = p_{023}$			⊕		⊕				
$a_1 \cdot b_1 = p_1$					⊕			⊕	
$a_3 \cdot b_3 = p_3$		⊕			⊕				
	c_8	c_7	c_6	c_5	c_4	c_3	c_2	c_1	c_0

Bei den neun Teil-Produkten aus **Tabelle 31** werden die Additionen (XOR) benötigt, um ihre Multiplikanden zu ermitteln, wobei alle Multiplikanden verschieden sind. Aus diesem Grund ist der minimale XOR-Aufwand der Berechnung der Eingänge der Teil-Multiplizierers 2·9=18 XOR-Gatter (bzw. 18m XOR-Gatter bei der 5-Segment-Polynome). Die optimierte Reihenfolge dieser Additionen (XOR) ist wie folgt:

$$
\begin{aligned}
&a_0 \oplus a_1 && b_0 \oplus b_1 \\
&a_0 \oplus a_4 && b_0 \oplus b_4 \\
&a_3 \oplus a_4 && b_3 \oplus b_4 \\
&(a_0 \oplus a_1) \oplus (a_3 \oplus a_4) = && (b_0 \oplus b_1) \oplus (b_3 \oplus b_4) = \\
&\quad = (a_0 \oplus a_1 \oplus a_3 \oplus a_4) && \quad = (b_0 \oplus b_1 \oplus b_3 \oplus b_4) \\
&(a_0 \oplus a_1 \oplus a_3 \oplus a_4) \oplus a_2 = && (b_0 \oplus b_1 \oplus b_3 \oplus b_4) \oplus b_2 = \\
&\quad = (a_0 \oplus a_1 \oplus a_2 \oplus a_3 \oplus a_4) && \quad = (b_0 \oplus b_1 \oplus b_2 \oplus b_3 \oplus b_4) \\
&(a_0 \oplus a_1 \oplus a_2 \oplus a_3 \oplus a_4) \oplus a_1 = && (b_0 \oplus b_1 \oplus b_2 \oplus b_3 \oplus b_4) \oplus b_1 = \\
&\quad = (a_0 \oplus a_2 \oplus a_3 \oplus a_4) && \quad = (b_0 \oplus b_2 \oplus b_3 \oplus b_4) \\
&(a_0 \oplus a_1 \oplus a_2 \oplus a_3 \oplus a_4) \oplus a_3 = && (b_0 \oplus b_1 \oplus b_2 \oplus b_3 \oplus b_4) \oplus b_3 = \\
&\quad = (a_0 \oplus a_1 \oplus a_2 \oplus a_4) && \quad = (b_0 \oplus b_1 \oplus b_2 \oplus b_4) \\
&(a_0 \oplus a_1 \oplus a_2 \oplus a_4) \oplus a_0 = && (b_0 \oplus b_1 \oplus b_2 \oplus b_4) \oplus b_0 = \\
&\quad = (a_1 \oplus a_2 \oplus a_4) && \quad = (b_1 \oplus b_2 \oplus b_4) \\
&(a_0 \oplus a_2 \oplus a_3 \oplus a_4) \oplus a_4 = && (b_0 \oplus b_2 \oplus b_3 \oplus b_4) \oplus b_4 = \\
&\quad = (a_0 \oplus a_2 \oplus a_3) && \quad = (b_0 \oplus b_2 \oplus b_3)
\end{aligned}
$$

(155)

Um die Montgomery-5-TD optimal zu berechnen, muss eine abstrakte TD (wie im Kapitel 3.9) untersucht werden, weil viele Zeilen der Montgomery-5-TD nur zwei ununterbrochen gefüllte Zellen haben und als Sonderfälle der Kriterien ***(K)*** zu

behandeln sind. **Tabelle 32** stellt diesen abstrakten Fall dar: sie besteht aus zwei Zeilen, p_k und p_s. Die Zeile p_k ist komplett gefüllt und bildet den iterativen TD-Teil. Die Zeile p_s besteht aus 2 Gruppen von je 2 ununterbrochen gefüllten Zellen. Zwischen diesen Gruppen befindet sich nur eine leere Zelle.

Tabelle 32: Abstrakte TD

p_k	⊕	⊕	⊕	⊕	⊕	⊕	⊕
p_s		⊕	⊕		⊕	⊕	
	c_6	c_5	c_4	c_3	c_2	c_1	c_0

Die iterative Berechnung der **Tabelle 32** nach dem Ablaufplan $c_0 \rightarrow c_1 \rightarrow \ ... \rightarrow c_6$ benötigt 2 XOR-Gatter pro Gruppe, d.h. 4 XOR-Gatter zusammen. D.h., dass bei der iterativen Berechnung der Spalte c_3 aus c_2 und der Spalte c_4 aus c_3 ein XOR-Gatter benötigt wird, um die Zelle (p_s, c_3) zu leeren und noch ein XOR-Gatter, um die Zelle (p_s, c_4) zu füllen. **Tabelle 33** zeigt die Änderungen, die in **Tabelle 32** vorgenommen wurden, um den XOR-Aufwand zu reduzieren.

Tabelle 33: Abstrakte TD

p_k	⊕	⊕	⊕	⊕	⊕	⊕	⊕	*iterativer* TD-Teil
p_s		⊕	⊕	⊕	⊕	⊕		
p_s				⊕				*separater* TD-Teil
	c_6	c_5	c_4	c_3	c_2	c_1	c_0	

Die leere Zelle (p_s, c_3), die in **Tabelle 32** leer ist, wird nun gefüllt. Um das zu kompensieren, wird eine weitere Zeile mit dem TP p_s und nur einer gefüllter Zelle (p_s, c_3) eingefügt. Der Wert $c_3 = p_k \oplus p_s \oplus p_s = p_k$ bleibt unverändert. Jetzt gehört aber die Zeile mit TP p_s und 5 ununterbrochen gefüllten Zellen eindeutig zu dem iterativen TD-Teil. Die neue Zeile mit TP p_s und nur einer gefüllten Zelle gehört zu dem *separaten* TD-Teil. Die Berechnung des *iterativen* TD-Teils nach dem gleichen Ablaufplan $c_0 \rightarrow c_1 \rightarrow \ ... \rightarrow c_6$ benötigt 2 XOR-Gatter. Noch ein XOR-Gatter wird benötigt, um die Werte von dem separaten TD-Teil zu addieren (XORen). Nur 3 XOR-Gatter werden insgesamt benötigt.

Solche Umwandlungen werden in **Tabelle 31** vorgenommen und in **Tabelle 34** gezeigt. Die nachgefüllten und kompensierenden Zellen sind in **Tabelle 34** grau gekennzeichnet.

Der XOR-Aufwand der optimalen Berechnung der Montgomery-5-TD (siehe **Tabelle 34**) setzt sich zusammen aus der:

- Berechnung der TM-Eingänge: 18 XOR-Gatter (siehe oben).
- Berechnung aller Spalten-Werte der TD:
  - 8 XOR-Gatter für die Berechnung des iterativen TD-Teils nach dem Ablaufplan $c_0^{it} \rightarrow ... \rightarrow c_5^{it}$; $c_8^{it} \rightarrow c_7^{it} \rightarrow c_6^{it}$
  - 17 XOR-Gatter für die Berechnung des separaten TD-Teils, der 19 gefüllte Zellen beinhaltet, weil die Summen ($p_0 \oplus p_{01}$) und ($p_4 \oplus p_{34}$) bereits bei der

Berechnung des iterativen TD-Teils ermittelt wurden und zur Verfügung stehen.

Tabelle 34: Umgewandelte **Tabelle 31**

p_0			⊕	⊕	⊕	**⊕**	⊕	⊕	⊕	
p_{01}				⊕	⊕	**⊕**	⊕	⊕		
p_{04}			⊕	⊕	**⊕**	⊕	⊕			*iterativer* TD-Teil
p_4	⊕	⊕	⊕	**⊕**	⊕	⊕	⊕			
p_{34}		⊕	⊕	**⊕**	⊕	⊕				
p_{01234}				⊕	⊕	⊕				
p_{0234}			⊕			⊕				
p_{0134}				⊕		⊕				
p_{0124}				⊕			⊕			
p_{124}					⊕		⊕			
p_{023}			⊕		⊕					
p_1					⊕			⊕		
p_3		⊕			⊕					*separater* TD-Teil
p_0						**⊕**				
p_{01}						**⊕**				
p_{04}					**⊕**					
p_4				**⊕**						
p_{34}				**⊕**						
	c_8	c_7	c_6	c_5	c_4	c_3	c_2	c_1	c_0	

So ist die Gatter-Komplexität der iterativ optimierten Montgomery-5-TD-Berechnung:

$$GC_5 = 13_{\&} + (18+8+17)_{XOR} = 13_{\&} + 43_{XOR}$$

(156)

Formel **(157)** stellt die Gatter-Komplexität des LP und die verursachte Signalverzögerung dar.

$$GC_{\substack{LP \\ 5-Bits}} = 1_{\&} + 7_{XOR}$$

$$delay_5^{Montgom} = T_{\&} + 7 \cdot T_{XOR}$$

(157)

Die Berechnung der TM-Eingänge nach der in [19] angegebener Reihenfolge (siehe **(152)**) verursacht die Steigerung des XOR-Aufwandes um 4 XOR-Gatter aber reduziert die Signalverzögerung um ein T_{XOR} und kann bei der Suche der optimalen Kombination der MM benutzt werden, wenn die Signalverzögerung ein weiterer kritischer Parameter der Schaltung ist.

Die Berechnung der Montgomery-5-Segment-TD wurde im Rahmen dieser Arbeit iterativ optimiert und ihre GC wurde im Algorithmus 2 für die Suche der optimalen Kombination der MM benutzt. Die Montgomery-5-Segment-TD wurde aus Montgomery-5-TD (siehe **Tabelle 34**) ermittelt. **Tabelle 35** zeigt die Aufteilung der Montgomery-5-Segment-TD in *iterativen* und *separaten* TD-Teil, wobei bei der *iterativen* Sub-TD die gleichgefüllten Zeilen bereits zusammen geXORt (Zeilen-Implosion) wurden.

Die Herleitung des XOR-Aufwandes der iterativ optimierten Berechnung der **Tabelle 35** wurde analog zu der Karatsuba-4-Segment-TD (siehe Kapitel 4.3.5) durchgeführt. Aus diesem Grund wird dies hier nicht mehr detailliert erklärt. Der Ablaufplan der Berechnung des iterativen TD-Teils so wie auch die XOR-Aufwände der Berechnungen der beiden TD-Teile sind in **Tabelle 35** gezeigt. Formel **(158)** zeigt die Gatter-Komplexität der iterativ optimierten Berechnung der Montgomery-5-Segment-TD und Formel **(159)** zeigt die GC des längsten Pfades und die verursachte Signalverzögerung dieser Berechnung.

Tabelle 35: Aufteilung der Montgomery-5-Segment-TD

	C_9	C_8	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0	
$TP_0[0]$				⊕	⊕	⊕	⊕	⊕	**⊕**	⊕	*iterativer* TD-Teil
$TP_0[1]$			⊕	⊕	⊕	⊕	⊕	⊕	⊕		
$TP_{01}[0]$					⊕	⊕	⊕	⊕	**⊕**		
$TP_{04}[0]⊕TP_{01}[1]$				⊕	⊕	⊕	⊕	⊕			((7+9)m-8)$_{XOR}$ für die Berechnung nach dem Ablaufplan:
$TP_{34}[0]⊕ TP_{04}[1]$			⊕	⊕	⊕	⊕	⊕				
$TP_{34}[1]$		**⊕**	⊕	⊕	⊕	⊕					
$TP_4[1]$	⊕	**⊕**	⊕	⊕	⊕	⊕	⊕				$C_0^{it} \rightarrow ... \rightarrow C_5^{it}$; $C_9^{it} \rightarrow ... \rightarrow C_6^{it}$
$TP_4[0]$		⊕	⊕	⊕	⊕	⊕	⊕	⊕			
$TP_{01234}[0]$					⊕	⊕	⊕				
$TP_{01234}[1]$				⊕	⊕	⊕					
$TP_0[0]$							**⊕**				*separater* TD-Teil
$TP_{01}[0]$							**⊕**				
$TP_{0234}[0]⊕ TP_{0124}[1]$				⊕			⊕				
$TP_{0134}[0]⊕ TP_{124}[1]$					⊕		⊕				
$TP_{0124}[0]⊕ TP_1[1]$					⊕			⊕			
$TP_4[0]$					⊕						
$TP_{34}[0]$					⊕						
$TP_{124}[0]$						⊕		⊕			
$TP_{023}[0]⊕ TP_{0134}[1]$				⊕		⊕					
$TP_1[0]$						⊕			⊕		(26m-8)$_{XOR}$
$TP_3[0]⊕ TP_{0234}[1]$			⊕			⊕					für die Addition (XOR) der TP-Segmente
$TP_{04}[0]$						⊕					
$TP_{023}[1]$			⊕		⊕						
$TP_3[1]$		⊕			⊕						
$TP_0[1]$						⊕					
$TP_{01}[1]$						⊕					
$TP_{04}[1]$					⊕						
$TP_4[1]$				**⊕**							
$TP_{34}[1]$				**⊕**							

$$GC_{5m}^{\substack{\textit{iterativ optimierte}\\ \text{Montgomery - MM}}} = 13 \cdot GC_m^{\substack{\text{MM des}\\ m\text{-TM}}} + \left(\underbrace{18m}_{\text{TM-Eingänge}} + \underbrace{7(m-1)}_{\text{Zeilen-Implosion}} + \underbrace{(16m-8)+(26m-8)}_{\text{Berechnung aller Spalten}} \right)_{XOR} =$$

$$= 13 \cdot GC_m^{\substack{\text{MM des}\\ m\text{-TM}}} + (67m-23)_{XOR}$$

(158)

$$GC_{5m}^{LP} = \text{GC}_{LP\ \text{des}\ m\text{-TM}} + 8_{XOR}$$

$$delay_{5m} = delay_m + 8 \cdot T_{XOR}$$

(159)

iterativ optimierte Montgomery - MM

MM des *m*-TM

iterativ optimierte Montgomery - MM

Kapitel 5

Optimale Kombination iterativ optimierter MM

In diesem Kapitel werden die Ergebnisse der Suche nach optimalen Kombinationen von Multiplikations-Methoden präsentiert.

Die optimale Kombination der MM wurde für alle Polynom-Längen bis 600 Bits mittels **Algorithmus 1** (Kapitel 3.5) ermittelt. **Tabelle 36** zeigt die 9 MM, die in dieser Arbeit bei der Bestimmung der optimalen Kombination von MM berücksichtigt wurden.

Tabelle 36: Untersuchte MM

Multiplikations-Methode	Abkürzung
klassische MM	klas
iterativ optimierte generalisierte Karatsuba-MM	io_genKar
iterativ optimierte n-Bits-Karatsuba-MM	ioKar
n-Bits-Karatsuba-MM mit Ersatz der Sub-TD	ioKar_mitErsatz
iterativ optimierte n-Segment-Karatsuba-MM	itK
iterativ optimierte n-Bits-Winograd-MM	ioWin
n-Bits-Winograd-MM mit Ersatz der Sub-TD	ioWin_mitErsatz
iterativ optimierte n-Segment-Winograd-MM	itW
iterativ optimierte 5-Segment-Montgomery-MM	ioMont_5

In der Literatur werden die Karatsuba- und die Winograd-MM normalerweise als eigenständige MM betrachtet. In dieser Arbeit wurden die iterativ optimierte Karatsuba- und die Winograd-MM nicht als eigenständige MM betrachtet, weil sie Sonderfälle der iterativ optimierten generalisierten Karatsuba-MM sind. Die rekursive Anwendung der untersuchten MM wurde ebenfalls nicht als eine eigenständige Multiplikations-Methode betrachtet. Wenn die rekursive Anwendung einer der MM optimal ist, wird dieses Ergebnis von dem **Algorithmus 1** gefunden.

In Kapitel 5.1 werden die ermittelten optimalen Kombinationen für die beiden IHP-Technologien[18] präsentiert und der Einfluss der Technologie auf die Ergebnisse des **Algorithmus 1** untersucht. Es wurde festgestellt, dass die flächen-optimale Kombinationen der MM für die 283- und 571-Bits langen Polynome auch energie-optimal sind. Dieses Ergebnis ist technologisch unabhängig
Die optimalen Kombinationen der MM werden für alle untersuchten Polynom-Längen im Kapitel 5.2 evaluiert. Kapitel 5.3 zeigt die Synthese-Ergebnisse der Multiplizierer für die ECC-relevanten Polynom-Längen.

5.1. Technologie-Abhängigkeit der optimalen MM-Kombinationen

Tabelle 37 zeigt für alle ECC-relevanten Polynom-Längen die Kombinationen der MM, die jeweils zur minimalen Fläche des Multiplizierers führen. Diese Ergebnisse wurden mittels **Algorithmus 1** für beide IHP-Technologien ermittelt.
Für jede Polynom-Länge *n*, die eine Primzahl ist, ist in **Tabelle 37** die optimale Kombination der MM für Polynome einer größeren Länge *n'* gezeigt. Die optimalen Kombinationen der MM sind als Segmentierungs-Hinweise beschrieben. Nach einer Zahl, die die Anzahl der Segmene zeigt, steht in Klammern der abgekürzte Name der für die Segmentierung verwendeten MM. Z.B. für die 163-Bits-Polynome ist die folgende Kombination der MM Flächen-optimal: 168=4(itK)·6(itW)·7(klas). Das bedeutet, dass für die Multiplikation der 163-Bits-Polynome der Multiplizierer für 168-Bits langen Operanden optimal wird, wobei:

- Die 168-Bits langen Polynome müssen in 4 Segmente nach der iterativ optimierten 4-Segment-Karatsuba-MM aufgeteilt werden.
- Jedes der 42-Bits langen Segmente (42=168/4) wird selber in 6 Teile unter der Anwendung der iterativ optimierten 6-Segment-Winograd-MM aufgeteilt.
- Jedes 7-Bits lange Segment (7=42/6) wird mit der klassischen MM in 7 1-Bits lange Segmente zerlegt.

Tabelle 37: Flächen-optimale Kombinationen der MM

n	n'	optimale Kombination MM	GC	delay	IHP 0,13µ		IHP 0,25µ	
					area, mm^2	energy, pJ	area, mm^2	energy, pJ
163	168=	4(itK)·6(itW)·7(klas)	$7938_{\&}$+ 11614_{XOR}	$1\cdot T_{\&}$ +$15\cdot T_{XOR}$	0.220	743.2946	1.074	686.0514
233	240=	2(itK)·4(itK)·6(itW)·5(klas)	$12150_{\&}$+ 20847_{XOR}	$1\cdot T_{\&}$ +$18\cdot T_{XOR}$	0.378	1257.69	1.838	1147.72
283	288=	4(itK)·4(itK)·6(itW)·3(klas)	$13122_{\&}$+ 29635_{XOR}	$1\cdot T_{\&}$ +$19\cdot T_{XOR}$	0.504	1636.564	2.437	1465.961
409	420=	7(itK)·2(itK)·6(itW)·5(klas)	$31050_{\&}$+ 55263_{XOR}	$1\cdot T_{\&}$ +$21\cdot T_{XOR}$	0.994	3291.761	-	-
	432=	4(itK)·4(itK)·9(itW)·3(klas)	26244_{AND}+58318_{XOR}	$1\cdot T_{\&}$ +$22\cdot T_{XOR}$	-	-	4.814	2901.648
571	576=	2(itK)·4(itK)·4(itK)·6(itW)·3(klas)	$39366_{\&}$+ 90918_{XOR}	$1\cdot T_{\&}$ +$22\cdot T_{XOR}$	1.540	4988.36	7.440	4461.896

[18] CMOS 0,25µ und CMOS 0,13µ

Tabelle 38 zeigt für alle ECC-relevanten Polynom-Längen die Kombinationen der MM, die den minimalen Energieverbrauch verursachen.

Tabelle 38: Energieverbrauchs-optimale Kombinationen der MM

n	n'	optimale Kombination MM	GC	delay	IHP 0,13µ		IHP 0,25µ	
					area, mm^2	energy, pJ	area, mm^2	energy, pJ
163	168=	7(itK)·2(itK)·4(itK) ·3(klas)	$5589_{AND}+13443_{XOR}$	$1\cdot T_{\&}+18\cdot T_{XOR}$	0.226	729.127	1.090	650.489
233	240=	5(itM)·4(itK)·4(itK)·3(klas)	$9477_{AND}+23213_{XOR}$	$1\cdot T_{\&}+20\cdot T_{XOR}$	0.388	1252.695	1.875	1116.306
283	288=	4(itK)·4(itK)·6(itW)·3(klas)	$13122_{AND}+29635_{XOR}$	$1\cdot T_{\&}+19\cdot T_{XOR}$	0.504	1636.564	2.437	1465.961
409	432=	4(itK)·4(itK)·9(itW)·3(klas)	$26244_{AND}+58318_{XOR}$	$1\cdot T_{\&}+22\cdot T_{XOR}$	0.996	3235.924	4.814	2901.648
571	576=	2(itK)·4(itK)·4(itK)·6(itW)·3(klas)	$39366_{AND}+90918_{XOR}$	$1\cdot T_{\&}+22\cdot T_{XOR}$	1.540	4988.360	7.440	4461.896

Der Einfluss der Technologie auf die Bewertung der MM-Kombinationen lässt sich über die Flächen- und Energie-Koeffizienten der AND- und XOR-Gatter beschreiben. Für beide IHP-Technologien sind die Flächen- und Energie-Koeffizienten (siehe Kapitel 3.2), die die Verhältnisse zwischen XOR- und AND-Gattern beschreiben, wie folgt:

$$k_A^{IHP(0{,}13\mu)} = 1{,}667 \qquad k_A^{IHP(0{,}25\mu)} = 1{,}500$$
$$k_E^{IHP(0{,}13\mu)} = 1{,}072 \qquad k_E^{IHP(0{,}25\mu)} = 0{,}797$$

(159)

Trotz des Unterschiedes bei den Flächen-Koeffizienten sind die optimalen Kombinationen der MM für die meisten ECC-relevanten Polynom-Längen für beide Technologien gleich. Dasselbe gilt für die energie-optimalen Kombinationen, wobei einige flächen-optimale Kombinationen[19] gleichzeitig energie-optimal sind (siehe **Tabelle 37** und **Tabelle 38**). Aus diesen Gründen wurden hier die optimalen Kombinationen der MM auch für andere Gatter-Verhältnisse mit Algorithmus 1 ermittelt, um den Einfluss der Technologie auf die Wahl der optimalen Kombinationen der MM zu untersuchen.

Algorithmus 1 wurde mit Koeffizienten-Werten aus der Bereich **(160)** mit dem Schritt $\Delta k=0{,}1$ initialisiert:

$$0{,}5 \le k \le 2{,}5$$

(160)

Ein XOR-Gatter besteht aus mehr Transistoren als ein AND-Gatter. Der Unterschied ist aber nicht mehr als 2 Mal [45], was auch die IHP-Technologie-Werte bestätigen. Damit deckt der Bereich **(160)** mehrere Technologien ab. Die optimalen Kombinationen der MM für verschiedene Technologie-Koeffizienten aus dem Bereich **(160)** sind in **Tabelle 39** gegeben. Falls der Koeffizient *k* einer Technologie nicht in dem Bereich **(160)** liegt, können die optimalen Kombinationen der MM mit dem entsprechend initialisierten **Algorithmus 1** ermittelt werden.

[19] Für die Polynom-Längen 283- und 571- Bits für IHP-Technologie 0,13µ; und für 283-, 409- und 571-Bits für IHP-Technologie 0,25µ

Nach dem Vergleich der theoretisch berechneten Flächen- und Energieverbrauchs-Werte der optimalen Multiplizierer aus **Tabelle 37** und **Tabelle 38** wurde folgendes festgestellt: Der Energieverbrauch bei den flächen- und energie-optimalen Kombinationen unterscheidet sich nur gering. Die Zeitverzögerung der Multiplizierer ist meistens bei den flächen-optimalen Kombinationen die kürzeste. Diese Tatsache kann bei der Wahl des Implementierungs-Kandidaten berücksichtigt werden.

Tabelle 39: Optimalen Kombinationen der MM für verschiedene Technologie-Koeffizienten

n	optimale Kombination MM	k
163	168=7(itK)·2(itK)·6(itW)·2(klas)	k = 0.5
	168=7(itK)·2(itK)·4(itK)·3(klas)	0.6 ≤ k ≤ 1.2
	168=4(itK)·6(itW)·7(klas)	1.3 ≤ k ≤ 2.5
233	240=5(itM)·4(itK)·6(itW)·2(klas)	k = 0.5
	240=5(itM)·4(itK)·4(itK)·3(klas)	0.6 ≤ k ≤ 1.1
	240=2(itK)·4(itK)·6(itW)·5(klas)	1.2 ≤ k ≤ 2.5
283	288=4(itK)·4(itK)·6(itW)·3(klas)	0.5 ≤ k ≤ 2.5
409	416=13(itK)·4(itK)·4(itK)·2(klas)	0.5 ≤ k ≤ 0.7
	432=4(itK)·4(itK)·9(itW)·3(klas)	0.8 ≤ k ≤ 1.5
	420=7(itK)·2(itK)·6(itW)·5(klas)	1.6 ≤ k ≤ 1.9
	420=5(itM)·2(itK)·6(itW)·7(klas)	2.0 ≤ k ≤2.5
571	576=2(itK)·4(itK)·4(itK)·6(itW)·3(klas)	0.5 ≤ k ≤ 2.5

5.2. Vergleich der theoretischen Ergebnisse

Für den Vergleich der im Rahmen dieser Arbeit ermittelten Ergebnisse wurden die rekursiv angewendete generalisierte und die „simple“ Karatsuba-MM[20] [18] aus folgenden Gründen gewählt:

- Der Artikel über die generalisierte Karatsuba-MM [18] wird am häufigsten als Vergleichspunkt gewählt.
- In [18] wurde bewiesen, dass die Komplexität der rekursiv angewendeten generalisierten MM für die folgende Zerlegung der Polynom-Länge n in Prim-Faktoren minimal ist: $nm = n_k \cdot n_{k-1} \cdot \ldots \cdot n_1 \cdot m \mid n_k \leq n_{k-1} \leq \ldots \leq n_1 \leq m$
- Die GC der rekursiv angewendeten generalisierten und die der „simplen“ Karatsuba-MM kann für beliebige Polynom-Längen berechnet werden.

Da für viele Polynom-Längen mit der „simplen“ Karatsuba-MM die kleinere Fläche erreicht wird, wurde jeweils die kleinere der beiden Flächen für jede Polynom-

[20] Als „simple“ Karatsuba-MM wird in [18] die originale Karatsuba-MM bezeichnet, die auch für die Multiplikation von Polynomen ungerader Länge verwendet wurde.

Länge als Vergleichs-Wert gewählt. In [18] ist die exakte GC der beiden Karatsuba Methoden für Polynom-Längen nur bis zur 128 Bits gegeben (siehe S.17 in [18]). Aus diesem Grund wurde die Komplexität dieser Methoden als exakte GC, die im Rahmen dieser Arbeit eingeführt und verwendet wurde, für beliebige Polynom-Längen hergeleitet.

5.2.1. Rekonstruierung der Vergleichs-Daten

Zuerst wird die GC der rekursiv angewendeten generalisierten Karatsuba-MM ermittelt.
Formel **(106)** wurde benutzt, um die Komplexität der rekursiven Verwendung der generalisierten Karatsuba Multiplikations-Methode zu ermitteln, wenn die Länge der Polynome wie folgt in Faktoren zerlegt werden kann: $nm = n_k \cdot n_{k-1} \cdot \ldots \cdot n_1 \cdot m$. In diesem Fall werden die Faktoren erst in n_k-Teile segmentiert, jedes dieser Segmente wird dann selbst in n_{k-1} Teile segmentiert usw. Die Zerlegung mit der generalisierten Karatsuba Multiplikations-Formel wird durchgeführt, bis die Segmente nur 1-Bits groß werden, d.h. mit $m=1$. Unter der Bedingung $\prod_{i=a}^{b|b<a} f(i) = 1$ zeigt Formel **(161)** die Komplexität der rekursiven Anwendung der generalisierten Karatsuba- Multiplikations-Methode.

$$\#Complexity^{rek.gen.KarMF}_{n_k n_{k-1} \ldots n_1 m} = \#Complexity_m \cdot \left(\prod_{i=1}^{k} \frac{n_i^{\,2} + n_i}{2} \right) +$$

$$= \left[\sum_{i=1}^{k} \left(\left(\prod_{j=i+1}^{k} \frac{n_j^{\,2} + n_j}{2} \right) \cdot \left(4m(n_i - 1) \left(\prod_{s=1}^{i} n_s \right) - \frac{3n_i^{\,2} - n_i}{2} + 1 \right) \right) \right]_{XOR} =$$

$$= \left[\prod_{i=1}^{k} \frac{n_i^{\,2} + n_i}{2} \right]_{\&} + \left[\sum_{i=1}^{k} \left(\left(\prod_{j=i+1}^{k} \frac{n_j^{\,2} + n_j}{2} \right) \cdot \left(4(n_i - 1) \left(\prod_{s=1}^{i} n_s \right) - \frac{3n_i^{\,2} - n_i}{2} + 1 \right) \right) \right]_{XOR}$$

(161)

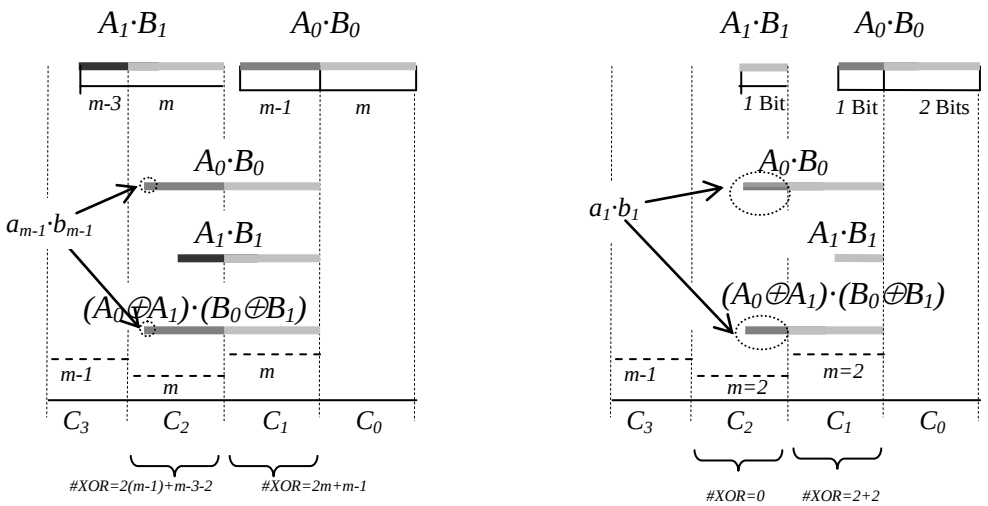
Mit der Formel **(161)** wurde die Komplexität für Polynom-Längen bis 600 Bits ermittelt. Für Polynom-Längen bis zu 128 Bits stimmt die berechnete Komplexität mit den Daten aus [18] überein (siehe Anhang 3).

Jetzt wird die GC der rekursiv angewendeten „simplen“ Karatsuba-MM diskutiert. Als „simple“ Karatsuba-MM ist in [18] die originale Karatsuba-MM bezeichnet, die auch für die Multiplikation von Polynomen ungerader Länge rekursiv verwendet wurde. Dafür werden die Polynome ungerader Länge in einen niedrigstwertigen und einen höchstwertigen Teil zerlegt, wobei der höchstwertige Teil um ein Bit größer ist als der niedrigstwertige. Das Produkt wurde in [18] als die Summe der 3

Teil-Produkte nach Karatsuba-MM dargestellt. Ein Teil-Produkt ist in diesem Fall um 2 Bits kleiner als die zwei anderen TP. Zum Beispiel: bei der Multiplikation der 3-Bits-Polynome wurde das Produkt als eine Summe zweier 3-Bits großer Teil-Produkte der 2-Bits-Polynome und eines 1-Bit großen Teil-Produktes der 1-Bit Polynome gesucht. Bei der Multiplikation der 7-Bits-Polynome wurde das Produkt als eine Summe zweier Teil-Produkte der 4-Bits-Polynome und eines Teil-Produktes der 3-Bit Polynome gesucht. Formel **(162)** zeigt die Multiplikation der Polynome ungerader Länge. Das niedrigstwertige Segment jedes 2-Segment-großen Operanden ist *m* Bits groß. Das höchstwertige Segment ist *(m-1)* Bits groß. Die Länge jedes TP ist in **(162)** gegeben. **Abbildung 24** illustriert Formel **(162)** und ihren Sonderfall für 3-Bits-Polynome, d.h. den Fall mit *m-1+m=3*.

$$A \cdot B = \underbrace{a_{2m-2} \ldots a_m}_{A_1} \underbrace{a_{m-1} \ldots a_0}_{A_0} \cdot \underbrace{b_{2m-1} \ldots b_m}_{B_1} \underbrace{b_{m-1} \ldots b_0}_{B_0} = (A_1 \cdot 2^m \oplus A_0) \cdot (B_1 \cdot 2^m \oplus B_0) =$$

$$= \underbrace{A_1 B_1}_{2m-3\,\text{Bits}} \cdot 2^{2m} \oplus \left(\underbrace{(A_0 \oplus A_1)(B_0 \oplus B_1)}_{2m-1\,\text{Bits}} \oplus A_1 B_1 \oplus A_0 B_0 \right) \cdot 2^m \oplus \underbrace{A_0 B_0}_{2m-1\,\text{Bits}} \qquad \textbf{(162)}$$



a) Polynom-Länge *(m-1)+m* Bits

b) Polynom-Länge *(m-1)+m=3* Bits

Abbildung 24: Graphische Darstellung der Karatsuba-MM für ungerade Polynom-Länge nach Segmentierung **(162)**.

Beim Vergleich mit der Multiplikation der Polynome gerader Länge wirkt sich die Tatsache, dass die Polynom-Segmente A_1 und B_1 um ein Bit kleiner sind als die Segmente A_0 und B_0, folgendermaßen auf die Gatter-Komplexität der Multiplikation aus:

- Die Ermittlung der Summen $(A_0 \oplus A_1)$ und $(B_0 \oplus B_1)$ benötigt nur 2(m-1) XOR-Gatter[21].
- Ein AND-Gatter bei der Berechnung der Teil-Produkte $A_0 \cdot B_0$ und $(A_0 \oplus A_1) \cdot (B_0 \oplus B_1)$ kann zusätzlich gespart werden, weil das höchstwertige Bit c_{2m-2} bei den beiden Teil-Produkten gleich $a_{m-1} \cdot b_{m-1}$ ist.
- Aus dem gleichen Grund können 2 XOR-Gatter bei der Berechnung der Spalte C_2 gespart werden (das höchstwertige Bit c_{2m-2} ist bei den Teil-Produkten $A_0 \cdot B_0$ und $(A_0 \oplus A_1) \cdot (B_0 \oplus B_1)$ gleich).

Die *separate* Berechnung der Segmente des Produktes C_i - die der Berechnung in [18] entspricht - hat unter der Berücksichtigung der oben genannten Ersparnisse die folgende Gatter-Komplexität:

$$\begin{aligned} \overset{Kar}{GC}_{m-1+m} = & \underbrace{GC_m}_{\substack{\text{Teil-Produkt} \\ A_0B_0}} + \underbrace{GC_{m-1}}_{\substack{\text{Teil-Produkt} \\ A_1B_1}} + \underbrace{\left(GC_m - 1_{\&}\right)}_{\substack{\text{Teil-Produkt} \\ (A_0 \oplus A_1)(B_0 \oplus B_1)}} + \\ & + \left[\underbrace{2(m-1)}_{\substack{\text{Berechnung} \\ A_0 \oplus A_1 \text{ und } B_0 \oplus B_1}} + \underbrace{(2m+m-1)}_{\text{Berechnung } C_1} + \underbrace{(2(m-1)+m-3-2)}_{\text{Berechnung } C_2} \right]_{XOR} = \\ & = 2 \cdot GC_m + GC_{m-1} - 1_{\&} + \left[8m-10\right]_{XOR} \end{aligned}$$

(163)

Formel **(163)** beschreibt die GC für jede ungerade Polynom-Länge m-1+m=2m-1 Bits, mit *m>1*, also auch für 3-Bits-Polynome.

Die GC der originalen Karatsuba-MM für Polynome gerader Länge kann aus der GC der generalisierten Karatsuba-MM (siehe **(106)**) mit *n=2* abgeleitet werden:

$$\overset{Kar}{GC}_{2m} = 3 \cdot GC_m + \left(8m-4\right)_{XOR}$$

(164)

Die GC der „simple“ Karatsuba-MM wurde mit **Algorithmus 6** für Polynom-Längen bis zu 600 Bits berechnet. **Algorithmus 6** wurde mit der GC für 1-Bit Polynome initialisiert.

[21] Im Fall der m-Bits großen Segmente A_1 und B_1 ist der XOR-Aufwand *2m* XOR-Gatter

Algorithmus 6

Input: Gate Complexity **(163)** and Gate Complexity **(164)**
Output: *GC(i)*, *1≤i≤n*
Initialization: $GC_1=1_{\&}+0_{XOR}$
Calculation
for *each i*|*3<i*≤n
if (*i* is odd **)**
m=(i+1)/2; calculation of $GC(i)=GC_{m-1+m}$ from **(163)**
else *m=i/2*; calculation of $GC(i)=GC_{2m}$ from **(164)**
end if
end for

Nach dem Vergleich der mit **Algorithmus 6** berechneten Daten wurde festgestellt, dass sie mit den Daten von [18] nicht übereinstimmen. Die Übereinstimmung wurde mit den folgenden Maßnahmen erreicht:

- mit der Initialisierung der GK der Multiplikation für 3-Bits-Polynome mit 6 AND- und 13 XOR-Gatter, was 1 XOR-Gatter weniger ist als tatsächlich notwendig wäre[22];
- mit der Berechnung der Komplexität aller anderen Polynome ungerader Länge *m-1+m>3* nach Formel **(165)**.

$$\overset{Kar}{GC}_{m-1+m} = \underbrace{GC_m}_{\substack{\text{Teil-Produkt}\\ A_0B_0}} + \underbrace{GC_{m-1}}_{\substack{\text{Teil-Produkt}\\ A_1B_1}} + \underbrace{\left(GC_m - 0_{\&}\right)}_{\substack{\text{Teil-Produkt}\\ (A_0\oplus A_1)(B_0\oplus B_1)}} +$$

$$+\left[\underbrace{2(m-1)}_{\substack{\text{Berechnung}\\ A_0\oplus A_1 \text{ und } B_0\oplus B_1}} + \underbrace{(2m+m-1)}_{\text{Berechnung } C_1} + \underbrace{(2(m-1)+m-3-0)}_{\text{Berechnung } C_2}\right]_{XOR} =$$

$$= 2\cdot GC_m + GC_{m-1} + \left[8m-8\right]_{XOR}$$

(165)

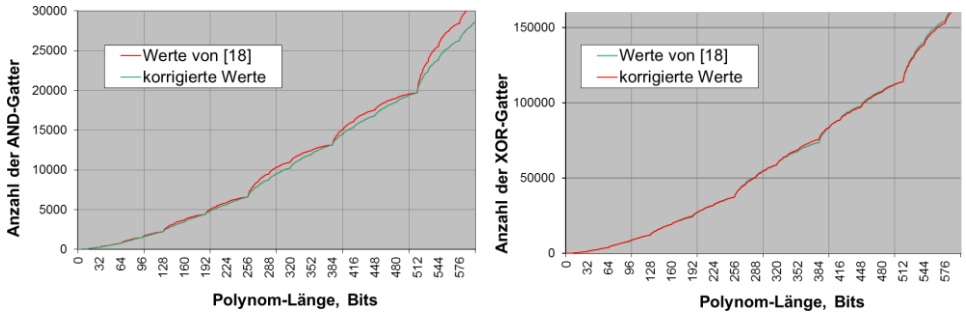
Der Unterschied der Formel **(165)** im Vergleich zur Formel **(163)** ist folgender: Die Einsparung eines AND- und zweier XOR-Gatter wurde in Formel **(165)** nicht berücksichtigt. Sie weist deshalb eine höhere GC aus als Formel **(163)**. [18] berücksichtigt die erwähnten Verbesserungen ebenfalls nicht. D.h. die GC aus **(165)** entspricht der aus [18].

Abbildung 25 zeigt die exakte Gatter-Komplexität beider „simplen“ Karatsuba-Multiplikationen: die rote Kurve zeigt die Daten, die mit den Werten von [18] übereinstimmen; die grüne Kurve entspricht den korrigierten Werten mit den richtigen Initialisierungs-Daten und mit der GC nach **(163)**.

Die durchschnittliche Reduktion der Anzahl der AND-Gatter im Vergleich zu den in [18] berichteten Daten beträgt 4,5%, maximal 8.03% bei einer Polynom-Länge von

[22] Nach Formel **(163)** sind 6 AND- und 14 XOR-Gatter notwendig

590 Bits. Insgesamt ist der AND-Aufwand bei 582 von 599 Polynom-Längen kleiner geworden.
Bei der Anzahl der XOR-Gatter gibt es negative wie auch positive Abweichungen von der roten Kurve. Die negative Abweichung zeigt, dass die korrigierten Werte kleiner sind als in [18]. Die positive Abweichung zeigt, dass die korrigierten Werte größer sind als in [18]. So ist der „neue“ XOR-Aufwand für 417 aus 599 Polynom-Längen kleiner geworden; wegen der Korrektur der Initialisierungs-Werte ist jedoch der „neue“ XOR-Aufwand für 166 von 599 Polynom-Längen gewachsen.



a) Anzahl der AND-Gatter b) Anzahl der XOR-Gatter

Abbildung 25: Vergleich der Gatter-Komplexität der „simplen“ Karatsuba-MM von [18] und der korrigierten Werte

Die Flächen, die aus der GC der beiden „simplen“ Karatsuba-MM berechnet wurden, sind in **Abbildung 26** gezeigt. Die rote Kurve basiert auf den Werten aus [18] und die grüne Kurve auf den korrigierten Werten.

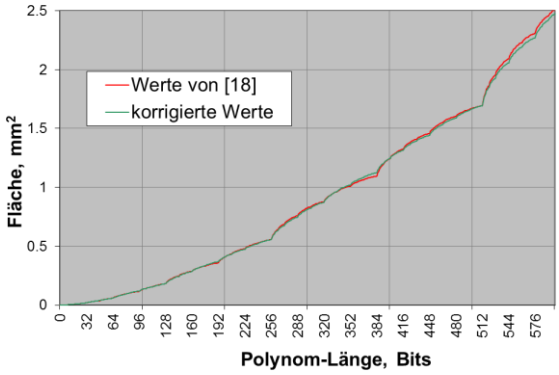


Abbildung 26: Vergleich der Flächen, die von der GC beider „simplen“ Karatsuba-MM berechnet wurden

Die grüne Kurve hat positive und negative Abweichungen von der roten Kurve (siehe **Abbildung 26**), die durchschnittlich etwa 1% betragen. Die positive Abweichung kommt bei 137 aus 599 Polynom-Längen vor und die negative bei 453 Polynom-Längen. Die maximalen Abweichungen sind +6% bei 3-Bits-Polynomen und -2% bei 9-Bits-Polynomen.

Die berechneten exakten GC der beiden „simplen“ rekursiven Karatsuba-MM, der rekursiven generalisierten Karatsuba-MM und der im Rahmen dieser Arbeit ermittelten optimalen Kombinationen der MM sind im Anhang 3 gegeben. Für den Vergleich der Ergebnisse dieser Arbeit wurden die besten Werte der beiden Karatsuba-MM verwendet: Für jede Polynom-Länge wurde die Fläche des Multiplizierers nach der korrigierten „simplen“ rekursiven Karatsuba-MM, und nach der rekursiven generalisierten Karatsuba-MM berechnet. Die beste der beiden wurde als Vergleich gewählt. Die rote Kurve in **Abbildung 27** stellt diese Werte dar. Die grüne Kurve zeigt die Ergebnisse dieser Arbeit, d.h. die berechneten Flächen der Multiplizierer bei optimaler Kombination der MM.

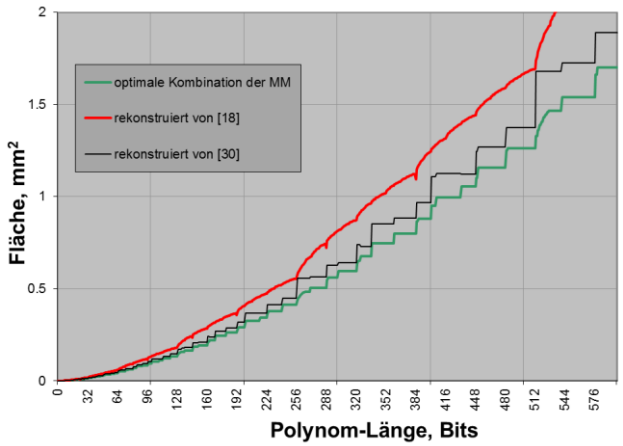


Abbildung 27: Vergleich der Flächen der Multiplizierer

Die Reduzierung der Fläche der optimalen Kombinationen der MM (grüne Kurve in **Abbildung 27**) im Vergleich zu den besten Werten beider rekursiven Karatsuba-Multiplikations-Methoden (rote Kurve in **Abbildung 27**) beträgt bei allen untersuchten Polynom-Längen von 2 bis zu 600 Bits durchschnittlich über 27%. Die maximale Abweichung von -43% wird bei 3-Bits-Polynomen und die minimale Abweichung von -22% wird bei den 452-Bits-Polynomen erreicht.

Die schwarze Kurve in **Abbildung 27** präsentiert die Ergebnisse aus [30]. Nach bestem Wissen der Autorin wurde eine optimale Kombination mehrerer MM nur in

[29] und [30] untersucht. In [29] wurden nur die originale Karatsuba-, die teilweise optimierte Winograd- und die klassische Multiplikations-Methode miteinander kombiniert. Die exakte GC und die optimale Segmentierung der Polynome bis 128 Bits ist in [29] gegeben. In [30] wurden die 5-, 6- und 7-Segment Montgomery Multiplikations-Methoden zu der Menge der untersuchten MM hinzugefügt. Die gesamte Anzahl der Gatter wurde in beiden Artikeln optimiert. Das Ziel war eine FPGA-Implementierung der Polynom-Multiplikation[23]. Die Daten aus [29] und aus [30] wurden im Rahmen dieser Arbeit rekonstruiert und miteinander verglichen. Die Ergebnisse aus [30] (schwarze Kurve in **Abbildung 27**) sind wegen der größeren Anzahl untersuchter MM besser als in [29]. **Tabelle 40** zeigt die GC der MM, die für die Rekonstruierung der Daten von [29] und [30] verwendet wurden. Die rekonstruierten Daten stimmen mit den in [29] angegebenen Daten für die Polynom-Längen bis 128 Bits überein.

Tabelle 40: Gatter-Komplexität der in [30] untersuchten MM

MM	Abkürzung laut [29] und [30]	GC
n- Segment-klassische	C_n	$\overset{C_n}{GC}_{nm} = n^2 \cdot GC_m + \left(2mn(n-1) - (n^2-1)\right)_{XOR}$
2-Segment-Karatsuba	K_2	$\overset{K_2}{GC}_{2m} = 3 \cdot GC_m + \left(8m-4\right)_{XOR}$
3-Segment-Winograd	K_3	$\overset{K_2}{GC}_{3m} = 6 \cdot GC_m + \left(22m-10\right)_{XOR}$
5-Segment-Montgomery	M_5	$\overset{M_5}{GC}_{5m} = 13 \cdot GC_m + \left(94m-40\right)_{XOR}$
6-Segment-Montgomery	M_6	$\overset{M_6}{GC}_{6m} = 17 \cdot GC_m + \left(150m-57\right)_{XOR}$
7-Segment-Montgomery	M_7	$\overset{M_7}{GC}_{7m} = 22 \cdot GC_m + \left(222m-80\right)_{XOR}$

Wenn die Multiplizierer nach der optimalen Kombination der MM (grüne Kurve in **Abbildung 27**) mit den Ergebnissen aus [30] (schwarze Kurve in **Abbildung 27**) verglichen werden, ergibt sich eine durchschnittliche Flächen-Reduktion von 9.6%.

Die Daten der grünen Kurve können mittels zweier Techniken weiter verbessert werden. Die beiden Techniken und die verbesserten Ergebnisse werden ausführlich im nächsten Abschnitt diskutiert. Die Anwendung dieser Techniken reduziert die Fläche der Multiplizierer durchschnittlich um weitere 2,6%. Die Signal-Verzögerung wächst aber wesentlich an, was die maximale Takt-Frequenz verringert.

[23] Ein FPGA (Field Programmable Gate Array) besteht aus mehreren gleichen Komponenten, deren Funktionalität als ein XOR- oder ein AND-Gatter programmiert werden kann.

5.2.2. Verbesserungs-Techniken

Der erste Verbesserungs-Ansatz zielt auf eine Glättung der Kurve, die die Fläche der Multiplizierer zeigt, ab. Hierfür wird vorgeschlagen einen *n*-Bit Multiplizierer auch für alle diejenigen *i*-Bits-Polynome, mit *i*<*n*, zu verwenden, für die gilt:

Fläche(n-Bits-Multiplizierer)<Fläche(i-Bits-Multiplizierer).

Diese Technik ist bereits in **Algorithmus 1** als Schritt 2 eingefügt. Die Anwendung dieser Technik im Schritt 1 des **Algorithmus 1** verbessert die Fläche der Multiplizierer durchschnittlich um 0.3%. **Algorithmus 7** zeigt die entsprechenden Änderungen im **Algorithmus 1**.

Algorithmus 7

Input: siehe **Algorithmus 1**
Output: siehe **Algorithmus 1**
Initialisierung: siehe **Algorithmus 1**

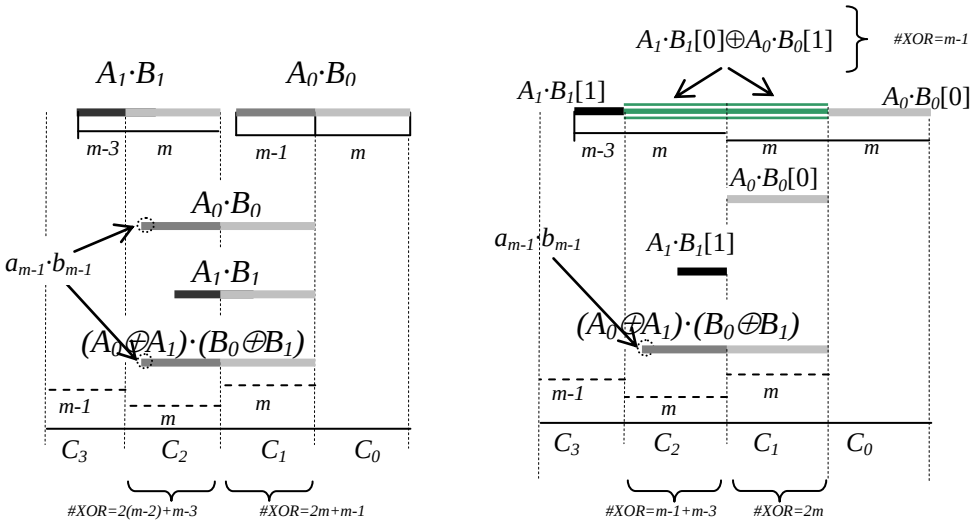
Berechnung
// Schritt 1
for $2 \le i \le n$ // siehe **Algorithmus 1**
...
• $opt\_GC_i = GC_i(MM\_opt[i])$ // siehe **Algorithmus 1**

// Anwendung der Verbesserungs-Technik aus ***Schritt 2***
for *each* $i|n>i \ge 1$
// siehe ***Schritt 2 des*** **Algorithmus 1**
end for
end for

Diese geringe Optimierung der Multiplizierer wird in **Abbildung 27** nicht dargestellt. Die exakten GC wurden berechnet und sind ebenfalls im Anhang 4 gegeben.

Die nächste Verbesserung nutzt die folgende Tatsache: die GC der 2-Segment-Karatsuba-MM für Polynome ungerader Länge ist kleiner als die der Polynome gerader Länge. Der Unterschied beträgt ein AND- und zwei XOR-Gatter. Diese Optimierung wurde bereits erklärt (siehe **Abbildung 24-a**). Das Kombinieren dieser Möglichkeit mit der iterativen Optimierung spart aber nur 1 AND-Gatter bei der Segmentierung der Operanden nach **(162)**. Bei der iterativen Berechnung wird das höchstwertige Segment des Teil-Produktes A_0B_0 mit dem niedrigstwertigen Segment des A_1B_1 addiert und diese Summe wird mit dem höchstwertigen Segment von $(A_0 \oplus A_1)\cdot(B_0 \oplus B_1)$ weiter addiert. In diesem Fall kann nur das Produkt $a_{m-1}b_{m-1}$, das bei der Ermittlung von A_0B_0 berechnet wurde, für die Ermittlung von

$(A_0 \oplus A_1)\cdot(B_0 \oplus B_1)$ wieder verwendet werden. **Abbildung 28** illustriert dieses. **Abbildung 28-a** wiederholt die **Abbildung 24-a**, die die separate Berechnung aller Produkt-Segmente zeigt. Die iterative Berechnung ist in **Abbildung 28-b** dargestellt. Formel **(166)** zeigt die exakte Gatter-Komplexität der iterativen Berechnung der Karatsuba Multiplikation mit der Segmentierung der Polynome ungerader Länge nach **(162).**



a) separate Berechnung der Produkt-Segmente b) iterative Berechnung der Produkt-Segmente

Abbildung 28: Illustration der Karatsuba-MM für Polynome ungerader Länge bei der Segmentierung entsprechend Formel **(162)**.

$$GC^{Kar}_{(m-1)+m} = \underbrace{GC_m}_{A_0B_0} + \underbrace{GC_{m-1}}_{A_1B_1} + \underbrace{\left(GC_m - 1_{\&}\right)}_{(A_0\oplus A_1)(B_0\oplus B_1)} + \left[\underbrace{2(m-1)}_{\substack{\text{Berechnung}\\ A_0\oplus A_1 \text{ und}\\ B_0\oplus B_1}} + \underbrace{(m-1)}_{\substack{\text{Berechnung}\\ A_1B_1[0]\oplus A_0B_0[1]}} + \underbrace{(2m+2m-4)}_{\substack{\text{Berechnung}\\ C_1 \text{ und} C_2}}\right]_{XOR} =$$

$$= 2\cdot GC_m + GC_{m-1} - 1_{\&} + \left[7m-7\right]_{XOR}$$

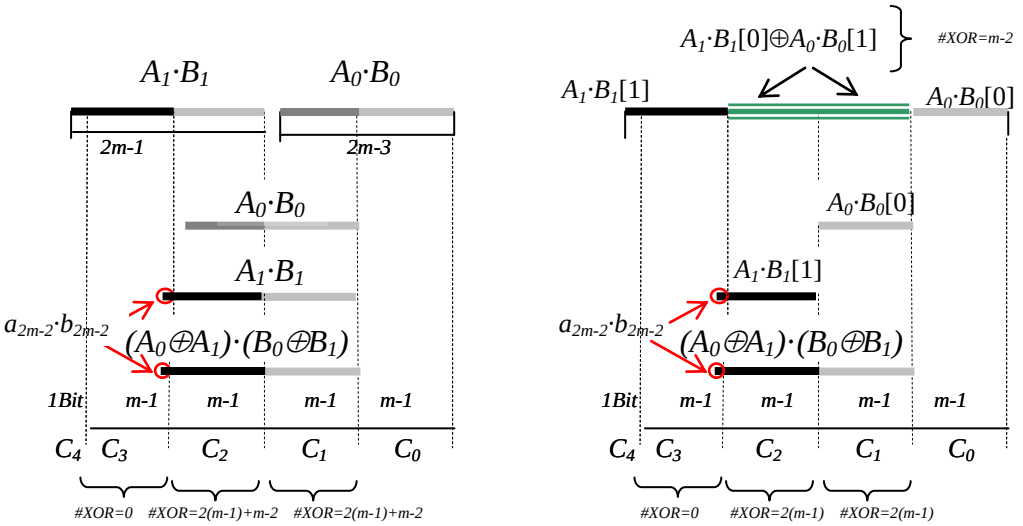
(166)

Ein besseres Ergebnis wird mit einer Segmentierung erreicht, bei der das höchstwertige Segment um 1 Bit größer ist als das niedrigstwertige. Formel **(167)** beschreibt diese Zerlegung, **Abbildung 29** illustriert sie.

$$A \cdot B = \underbrace{a_{2m-2} \ldots a_{m-1}}_{A_1} \underbrace{a_{m-2} \ldots a_0}_{A_0} \cdot \underbrace{b_{2m-1} \ldots b_{m-1}}_{B_1} \underbrace{b_{m-2} \ldots b_0}_{B_0} = (A_1 \cdot 2^{m-1} \oplus A_0)(B_1 \cdot 2^{m-1} \oplus B_0) =$$

$$= \underbrace{A_1 B_1}_{2m-1\,\text{Bits}} \cdot 2^{2m-2} \oplus \left(\underbrace{(A_0 \oplus A_1)(B_0 \oplus B_1)}_{2m-1\,\text{Bits}} \oplus A_1 B_1 \oplus A_0 B_0 \right) \cdot 2^{m-1} \oplus \underbrace{A_0 B_0}_{2m-3\,\text{Bits}}$$

(167)



a) separate Berechnung der Produkt-Segmente b) iterative Berechnung der Produkt-Segmente

Abbildung 29: Illustration der Karatsuba-MM für Polynome ungerader Länge bei Segmentierung entsprechend Formel **(167)**.

Diese Zerlegungs-Technik wurde in [49] veröffentlicht. Auch die Möglichkeit, die Produkt-Segmente iterativ zu berechnen, wurde in [49] benutzt. Die Ersparnis eines AND-Gatters wurde jedoch nicht berücksichtigt bzw. erwähnt.

Die Formeln **(168)** und **(169)** zeigen die exakte Gatter-Komplexität der beiden Berechnungen der Karatsuba-Multiplikation unter der Berücksichtigung der Einsparung eines AND- und zweier XOR-Gatter, vgl. auch **Abbildung 29**.

$$GC^{Kar,separat}_{m+(m-1)} = \underbrace{GC_{m-1}}_{A_0 B_0} + \underbrace{GC_m}_{A_1 B_1} + \underbrace{(GC_m - 1_{\&})}_{(A_0 \oplus A_1)(B_0 \oplus B_1)} + \left[\underbrace{2(m-1)}_{\substack{\text{Berechnung} \\ A_0 \oplus A_1 \text{ und} \\ B_0 \oplus B_1}} + \underbrace{((3m-4)+(3m-4))}_{\substack{\text{Berechnung} \\ C_1 \text{ und} C_2}} \right]_{XOR} =$$

$$= 2 \cdot GC_m + GC_{m-1} - 1_{\&} + [8m-10]_{XOR}$$

(168)

$$GC^{Kar,iterativ}_{m+(m-1)} = \underbrace{GC_{m-1}}_{A_0B_0} + \underbrace{GC_m}_{A_1B_1} + \underbrace{\left(GC_m - 1_{\&}\right)}_{(A_0 \oplus A_1)(B_0 \oplus B_1)} +$$

$$+ \left[\underbrace{2(m-1)}_{\substack{\text{Berechnung} \\ A_0 \oplus A_1 \text{ und} \\ B_0 \oplus B_1}} + \underbrace{(m-2)}_{\substack{\text{Berechnung} \\ A_1B_1[0] \oplus A_0B_0[1]}} + \underbrace{((2m-2)+(2m-2))}_{\substack{\text{Berechnung} \\ C_1 \text{ und} C_2}} \right]_{XOR} =$$

$$= 2 \cdot GC_m + GC_{m-1} - 1_{\&} + \left[7m - 8\right]_{XOR}$$

(169)

Aus dem Vergleich der Gatter-Komplexität der separaten und iterativen Berechnungen der Karatsuba Multiplikation (siehe **(162)** und **(167)**) ergibt sich folgendes:

- Die GC der separaten Berechnung ist in beiden Fällen gleich
- Die GC der iterativen Berechnung ist bei der Segmentierung **(167)** um 1 XOR-Gatter besser im Vergleich mit der Segmentierung nach **(162)**
- Die iterative Berechnung bei Segmentierung nach **(167)** ist für jede Länge der Segmente m>2 günstiger als die separate Berechnung.

Im Rahmen dieser Arbeit wurde die iterative Karatsuba Multiplikation für Polynome ungerader Länge mit der Segmentierungs-Technik **(167)** als eine eigenständige MM (io_uK) in die Untersuchungen einbezogen. Ihre Gatter-Komplexität ergibt sich aus **(169)**. Die Anwendung der beiden Verbesserungs-Techniken reduziert die Fläche der Multiplizierer durchschnittlich um 2,6%. Die blaue Kurve in **Abbildung 30** stellt diese Werte dar. Die exakten GC sind ebenfalls im Anhang 4 gegeben. Alle anderen Kurven in **Abbildung 30** stimmen mit den Kurven aus **Abbildung 27** überein.

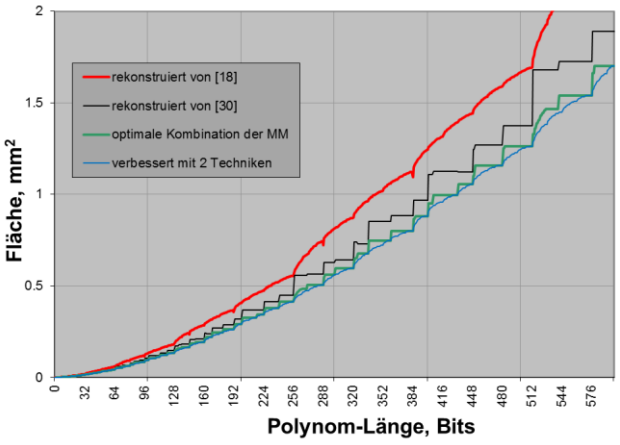


Abbildung 30: Flächen der Multiplizierer: die blaue Kurve zeigt die mit beiden Verbesserungs-Techniken erreichten Werte für die 0,13µ-IHP-Technologie.

Die durchschnittliche Abweichung der verbesserten Flächen-Werte (blaue Kurve in **Abbildung 30**) von den Werten in [30] (schwarze Kurve) beträgt 12% und kommt bei 593 aus den 599 untersuchten Polynom-Längen vor. Die maximale Abweichung -23,7% ergibt sich bei den 257-Bits langen Polynomen. Bei den ECC-relevanten Polynom-Längen beträgt die durchschnittliche Reduktion der Fläche der Multiplizierer 12%.

Tabelle 41 und **Tabelle 42** zeigen die GC und die hieraus berechneten Flächen der Multiplizierer und die maximalen Taktraten für alle ECC-relevanten Größen.

Tabelle 41: Parameter verschiedener 163-Bits- Polynom-Multiplizierer

n	Chip-Parameter		Data von [18]	Data von [30]	Data von [49]	Optimum der 9 MM	Optimum der 10 MM, mit io_uK
163	GC	#AND	3600	7938	6779	7938	6145
		#XOR	20298	12860	11623	11614	11899
	delay, T_{XOR}		24	16	18	15	19
	Fläche, mm^2		0.302	0.237	0.211	0.220	0.210
	F_{max}, MHz		459	674	603	716	573

Tabelle 42: Parameter verschiedener Polynom-Multiplizierer

n	Chip-Parameter		Data von [18]	Data von [30]	Optimum der 9 MM	Optimum der 10 MM, mit io_uK
233	GC	#AND	5997	12150	12150	12067
		#XOR	34120	23516	20847	20445
	delay, T_{XOR}		24	19	18	19
	Fläche, mm^2		0.507	0.414	0.378	0.372
	F_{max}, MHz		459	573	603	573
283	GC	#AND	8617	13122	13122	13041
		#XOR	49440	34148	29635	29502
	delay, T_{XOR}		27	21	19	20
	Fläche, mm^2		0.734	0.565	0.504	0.502
	F_{max}, MHz		410	521	573	546
409	GC	#AND	15093	26244	31050	29270
		#XOR	87288	67988	55263	52828
	delay, T_{XOR}		27	22	21	23
	Fläche, mm^2		1.296	1.126	0.994	0.947
	F_{max}, MHz		410	499	521	478
571	GC	#AND	26102	39366	39366	39123
		#XOR	152046	104744	90918	90484
	delay, T_{XOR}		30	24	22	23
	Fläche, mm^2		2.256	1.726	1.540	1.533
	F_{max}, MHz		370	459	499	478

Wie es in **Tabelle 41** und in **Tabelle 42** zu sehen ist, haben die optimalen Kombinationen aus 10 MM die minimalen Flächen. Die minimale Signal-Verzögerung haben aber die optimalen Kombinationen aus 9 MM, d.h ohne Methode io_uK.

5.2.3. Evaluierung der theoretischen Ergebnisse

Die Evaluierung der theoretischen Ergebnisse mit Synthese Daten wurde im Rahmen dieser Arbeit für die optimalen Kombinationen der klassischen MM mit iterativ optimierten *n*-Segment-Karatsuba und Winograd-MM, mit $2 \le n \le 4$, durchgeführt.
Die Multiplizierer wurden mit Design Vision von Synopsis [54] für IHP-Technologien 0,13µ und 0,25µ synthetisiert. **Tabelle 43** zeigt die optimalen Kombinationen dieser MM für die ECC-relevanten Polynom-Längen sowie auch ihre GC, die theoretisch berechneten und die synthetisierten Flächen der Multiplizierer. **Tabelle 44** zeigt die Signal-Verzögerungen der Multiplizierer. **Abbildung 31** stellt die berechneten Chip-Parameter der Multiplizierer und die Synthese-Ergebnisse graphisch dar. Auch die Parameter zweier weiterer Multiplizierer sind zum Vergleich angegeben: der 256-Bits-Multiplizierer stellt die Kombination der iterativ optimierten 2-Segment-Karatsuba-MM mit der klassischen MM dar. Die iterativ optimierte 2-Segment-Karatsuba-MM wird rekursiv angewendet, bis die Teil-Operanden 4 Bits groß sind. Diese werden dann mit der klassischen MM multipliziert. Die Synthese-Tools bieten auch gewisse Optimierungen an. Z.B. wenn einer der Eingänge eines AND-Gatters immer Null gesetzt ist, muss hier kein AND-Gatter benutzt werden, sondern das entsprechende Ausgangs-Signal kann Null gesetzt werden. Die Synthese-Ergebnisse für den 233-Bits-Multiplizierer bestätigen die Vermutung hinsichtlich der Optimierung durch die Synthese-Werkzeuge. D.h., der 233-Bits-Multiplizierer wurde aus dem 256-Bits-Multiplizierer durch Nullsetzen der höchstwertigen 23 Bits der beiden Operanden entworfen. Seine Fläche ist für die beiden IHP-Technologien kleiner die des 256-Bits-Multiplizierers.

Tabelle 43: Flächen der Multiplizierer

n	n´	#AND(n')	#XOR(n')	Fläche, mm^2			
				0,25µ		0,13µ	
				theor. für n´	Synthese für n	theor. für n´	Synthese für n
256	256=2·2·2·2·2·2·2·4	11664	24089	2.0236	1.6985	0.4181	0.3871
233 aus 256	256=2·2·2·2·2·2·2·4	11664	24089	2.0236	1.6956	0.4181	0.3865
163 aus 168	168=2·4·3·7	7938	11766	1.0833	0.8697	0.2223	0.2033
233 aus 240	240=4·4·3·5	12150	21100	1.8543	1.5022	0.3818	0.3552
283 aus 288	288=2·4·4·3·3	13122	30105	2.4673	2.0831	0.5108	0.4847
409 aus 432	432=4·4·3·3·3	26244	59452	4.8865	4.1290	1.0114	0.9613
571 aus 576	576=4·4·4·3·3	39366	92185	7.5207	6.3417	1.5575	1.4832

Tabelle 44: Signal-Verzögerungen der Multiplizierer

n	n´	delay	Signalverzögerung			
			0,25µ		0,13µ	
			theor. für n´	Synthese für n	theor. für n´	Synthese für n
256	256=2·2·2·2·2·2·4	$1 \cdot T_{AND} + 20 \cdot T_{XOR}$	10.51	6.81	1.83	3.2
233 aus 256	256=2·2·2·2·2·2·4	$1 \cdot T_{AND} + 20 \cdot T_{XOR}$	10.51	6.81	1.83	3.2
163 aus 168	168=2·4·3·7	$1 \cdot T_{AND} + 15 \cdot T_{XOR}$	7.96	5.87	1.40	2.74
233 aus 240	240=4·4·3·5	$1 \cdot T_{AND} + 17 \cdot T_{XOR}$	8.98	5.98	1.57	3.05
283 aus 288	288=2·4·4·3·3	$1 \cdot T_{AND} + 19 \cdot T_{XOR}$	10	6.46	1.74	3.20
409 aus 432	432=4·4·3·3·3	$1 \cdot T_{AND} + 20 \cdot T_{XOR}$	10.51	7.13	1.83	3.52
571 aus 576	576=4·4·4·3·3	$1 \cdot T_{AND} + 21 \cdot T_{XOR}$	11.02	7.65	1.92	3.77

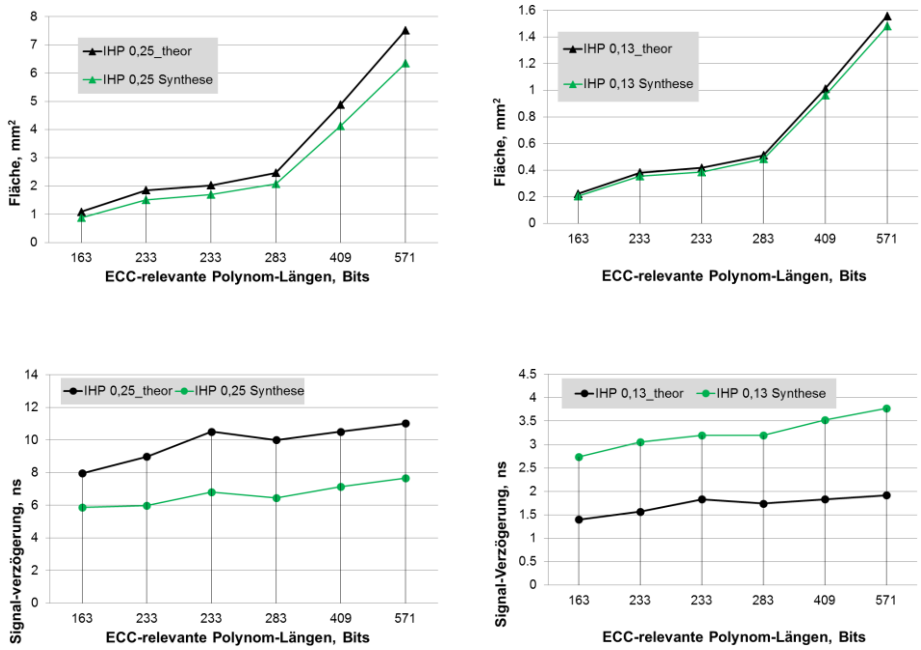


Abbildung 31: Chip-Parameter der Multiplizierer für ECC-relevante Polynom-Längen: berechnete Werte und Synthese-Daten

Die Flächen der synthetisierten Multiplizierer sind kleiner als die theoretischen Werte. Der Unterschied beträgt durchschnittlich 17% für die IHP-0,25µ-Technologie und 6% für die IHP-0,13µ-Technologie. Diese Tatsache kann folgendermaßen erklärt werden: die Multiplizierer wurden nicht nur mit AND- und XOR-Gatter mit 2 Eingängen implementiert, wie es bei der GC berechnet wurde, sondern auch mit anderen Gattern, die in den Technologien zur Verfügung stehen, z.B. mit XOR-Gattern mit 3 Eingängen. Für die IHP-0,25µ-Technologie ist die Abweichung der praktischen und theoretischen Flächen-Werte wesentlich größer

als für die IHP-0,13µ-Technologie. Diese Tatsache ist durch die anderen Basis-Elemente erklärbar. Die 0,25µ-Technologie hat keine AND- und XOR-Gatter, sondern NAND- und XNOR-Gatter. Die Funktionalität, die Fläche und die Signal-Verzögerung eines AND-Gatters (bzw. eines XOR-Gatters) wurde im Kapitel 3.3. mittels eines NAND-Gatters (bzw. XNOR-Gatters) und eines Invertors ermittelt. Aufgrund von der Eigenschaft der XOR-Funktion $a \oplus b = \overline{a} \oplus \overline{b}$ können bei der Implementierung in der 0,25µ-Technologie an mehreren Stellen nun NAND- und XNOR-Gatter, ohne Invertoren, benutzt werden.

Die Benutzung der Gatter mit gleicher Funktionalität aber mit anderer Ausgangsleistung erklärt den Unterschied bei den erwarteten und den synthetisierten Signal-Verzögerungs-Werten der Multiplizierer für die IHP-0,13µ-Technologie, insbesondere einer höheren Ausgangsleistung als der für die Berechnung verwendeten. Der Unterschied beträgt durchschnittlich 90% (siehe **Tabelle 44**). Für die IHP-0,25µ-Technologie ist die durchschnittliche Signal-Verzögerung um 30% kleiner als die erwartete, was wieder durch die Benutzung der NAND- und XNOR-Gatter, ohne Invertoren, erklärt werden kann.

Um die erwartete Reduktion der Chip-Fläche der optimalen Multiplizierer zu evaluieren, wurden die Flächen-Verhältnisse für jeden Multiplizierer für ECC-relevante Polynom-Längen und für den 256-Bits-Multiplizierer (siehe **Tabelle 43**) berechnet. Die Fläche des 256-Bits-Multiplizierers wurde für die Normierung verwendet. **Tabelle 45** zeigt diese Verhältnisse.

Tabelle 45: Flächen-Verhältnisse der Multiplizierer

n	$\alpha = \frac{Fläche(n-Bits \quad Multiplizierer)}{Fläche(256-Bits \quad Multiplizierer)}$			
	0,25µ		0,13µ	
	theor. Daten	Synthese-Daten	theor. Daten	Synthese-Daten
256	1	1	1	1
163 aus 168	0.5	0.5	0.5	0.5
233 aus 240	0.9	0.9	0.9	0.9
283 aus 288	1.2	1.2	1.2	1.3
409 aus 432	2.4	2.4	2.4	2.5
571 aus 576	3.7	3.7	3.7	3.8

Die Flächen-Verhältnisse der synthetisierten Multiplizierer bestätigen die erwartete Flächen-Reduktion (siehe **Tabelle 45**).

Kapitel 6

Optimierter serieller Multiplizierer

In diesem Kapitel werden die Simulations- und die Synthese-Ergebnisse für unterschiedliche Multiplizierer diskutiert. Insbesondere fokussiert sich das Kapitel auf die Multiplikation für 233-Bits-Polynome, weil diese Länge der Operanden für den Zeitraum 2011 – 2030 den Sicherheits-Vorschriften [13] entspricht.
In [27], [48] und [28] wurden serielle 233-Bits-Multiplizierer auf Basis der Karatsuba MM veröffentlicht. In [28] wurde eine verbesserte Struktur des Multiplizierers präsentiert, bei der mittels einer mehrfachen Nutzung derselben XOR-Gatter eine wesentliche Verbesserung der Chip-Parameter erreicht wurde.
In diesem Kapitel wird die optimierte Struktur des seriellen (Mehrtakt) Multiplizierers diskutiert und ihre Gatter-Komplexität ermittelt. Die im Kapitel 5 ermittelten optimalen kombinatorischen Teil-Multiplizierer können jetzt verwendet werden. Das reduziert die Fläche- und den Energieverbrauch des 233-Bits-Muiltplizierers. Im Gegensatz zu [27] und [28] wird in diesem Kapitel ein Design unter Verwendung des optimalen Teil-Multiplizierers für Polynome deutlich kleinerer Länge verwendet.
Die optimierte Struktur der seriellen Multiplizierer wird in Kapitel 6.1 diskutiert. Die Simulations- und die Synthese-Ergebnisse für serielle Multiplizierer, die nur einen Teil-Multiplizierer beinhalten, werden im Kapitel 6.2 präsentiert.

6.1. Struktur des seriellen Multiplizierers

Die serielle Ausführung einer Reihe gleicher Operationen (z.B. der Multiplikationen der Polynom-Segmente oder der XOR-Operationen der Teil-Produkte) ermöglicht die mehrfache Nutzung derselben Gatter in einer Schaltung. Dadurch kann die Chip-Fläche des Multiplizierers wesentlich reduziert werden. Zusätzlicher Aufwand entsteht aus der Fläche und aus dem Energie-Verbrauch der Einheiten, die die notwendigen Werte zu den Eingängen der mehrfach benutzten Schaltungs-Einheiten liefert.

Tabelle 46 zeigt den taktgesteuerten Ablaufplan der Multiplikation der 4-Segment großen Polynome entsprechend der Tabellarischen Darstellung der generalisierten Karatsuba-Multiplikations-Formel. Mit jedem Takt steht nur ein Teil-Produkt zur Verfügung. Bei jedem der Produkt-Segmente kann nur einer der folgenden Fälle auftreten:
1 – das niedrigstwertige Segment TP[0] des Teil-Produktes ist zu akkumulieren
2 – das höchstwertige Segment TP[1] des Teil-Produktes ist zu akkumulieren
3 – die Summe (XOR) der beiden TP-Segmente TP[0] ⊕TP[1] ist zu akkumulieren
4 – keine Werte sind zu akkumulieren, d.h. der Null-Wert ist zu akkumulieren

So wird z.B. im ersten Takt TP[0] in C_0 akkumuliert, das höchstwertige Segment TP[1] in C_4 und die Summe der beiden TP-Segmente in C_1, C_2 und C_3. Bei den anderen Produkt-Segmenten, d.h. in C_5, C_6 und C_7, werden in diesem Takt keine Werte akkumuliert.

Tabelle 46: genKar-4-Segment-TD

Takt-Nr.	Akkumulation der Teil-Produkt-Segmente									
	Gelieferte TP-Segmente	Schematische Darstellung der Akkumulation								Ausführungs-Hinweise
1	$A_0 \cdot B_0[0]=TP_0[0]$					⊕	⊕	⊕	⊕	$C_0=TP[0]$, $C_4=TP[1]$, $C_i=TP[0] \oplus TP[1], 0<i<4$ $C_i=0, i>4$
	$A_0 \cdot B_0[1]=TP_0[1]$				⊕	⊕	⊕	⊕		
2	$A_1 \cdot B_1[0]=TP_1[0]$				⊕	⊕	⊕	⊕		$C_1=C_1 \oplus TP[0]$, $C_5=C_5 \oplus TP[1]$, $C_i=C_i \oplus TP[0]+TP[1], 1<i<5$ $C_i=C_i \oplus 0, i=\{0,6,7\}$
	$A_1 \cdot B_1[1]=TP_1[1]$			⊕	⊕	⊕	⊕			
3	$A_2 \cdot B_2[0]=TP_2[0]$			⊕	⊕	⊕	⊕			$C_2=C_2 \oplus TP[0]$, $C_6=C_6 \oplus TP[1]$, $C_i=C_i \oplus TP[0]+TP[1], 2<i<6$ $C_i=C_i \oplus 0, i=\{0,1,7\}$
	$A_2 \cdot B_2[1]=TP_2[1]$		⊕	⊕	⊕	⊕				
4	$A_3 \cdot B_3[0]=TP_3[0]$		⊕	⊕	⊕	⊕				$C_3=C_3 \oplus TP[0]$, $C_7=C_7 \oplus TP[1]$, $C_i=C_i \oplus TP[0]+TP[1], 3<i<7$ $C_i=C_i \oplus 0, i=\{0,1,2\}$
	$A_3 \cdot B_3[1]=TP_3[1]$	⊕	⊕	⊕	⊕					
5	$(A_0 \oplus A_1) \cdot (B_0 \oplus B_1)$ [0] $=TP_{01}[0]$							⊕		$C_1=C_1 \oplus TP[0]$, $C_2=C_2 \oplus TP[1]$, $C_i=C_i \oplus 0, i=\{0,3,4,5,6,7\}$
	$(A_0 \oplus A_1) \cdot (B_0 \oplus B_1)$ [1] $=TP_{01}[1]$						⊕			
6	$(A_0 \oplus A_2) \cdot (B_0 \oplus B_2)$ [0]$=TP_{02}[0]$						⊕			$C_2=C_2 \oplus TP[0]$, $C_3=C_3 \oplus TP[1]$, $C_i=C_i \oplus 0, i=\{0,1,4,5,6,7\}$
	$(A_0 \oplus A_2) \cdot (B_0 \oplus B_2)$ [1]$=TP_{02}[1]$					⊕				
7	$(A_0 \oplus A_3) \cdot (B_0 \oplus B_3)$ [0]$=TP_{03}[0]$					⊕				$C_3=C_3 \oplus TP[0]$, $C_4=C_4 \oplus TP[1]$, $C_i=C_i \oplus 0, i=\{0,1,2,5,6,7\}$
	$(A_0 \oplus A_3) \cdot (B_0 \oplus B_3)$ [1]$=TP_{03}[1]$				⊕					
8	$(A_1 \oplus A_2) \cdot (B_1 \oplus B_2)$ [0]$=TP_{12}[0]$					⊕				$C_3=C_3 \oplus TP[0]$, $C_4=C_4 \oplus TP[1]$, $C_i=C_i \oplus 0, i=\{0,1,2,5,6,7\}$
	$(A_1 \oplus A_2) \cdot (B_1 \oplus B_2)$ [1]$=TP_{12}[1]$				⊕					
9	$(A_1 \oplus A_3) \cdot (B_1 \oplus B_3)$ [0]$=TP_{13}[0]$				⊕					$C_4=C_4 \oplus TP[0]$, $C_5=C_5 \oplus TP[1]$, $C_i=C_i \oplus 0, i=\{0,1,2,3,6,7\}$
	$(A_1 \oplus A_3) \cdot (B_1 \oplus B_3)$ [1]$=TP_{13}[1]$			⊕						
10	$(A_2 \oplus A_3) \cdot (B_2 \oplus B_3)$ [0]$=TP_{23}[0]$			⊕						$C_5=C_5 \oplus TP[0]$, $C_6=C_6 \oplus TP[1]$, $C_i=C_i \oplus 0, i=\{0,1,2,3,4,7\}$
	$(A_2 \oplus A_3) \cdot (B_2 \oplus B_3)$ [1]$=TP_{23}[1]$		⊕							
		C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0	

Für den ganzen Zyklus der Produkt-Ermittlung, der in unserem Beispiel 10 Takte benötigt, wird bei jedem der Segmente C_i, $1<i<2n-2$, mindestens einmal ein TP-Segment (TP[0] oder TP[1]), einmal die Summe der TP-Segmente (TP[0]⊕TP[1]) und einmal der Null-Wert akkumuliert. **Abbildung 32** zeigt die Struktur einer

Akkumulations-Einheit für ein Produkt-Segment C_i, die diese Möglichkeiten berücksichtigt.
Diese Struktur ist auch für die Akkumulation in den Segmenten C_1 und C_{2n-2} optimal, obwohl bei jedem dieser Segmente nur 3 Fälle auftreten können:

- bei Segment C_1: 1) der Null-Wert, 2) das Segment TP[0], 3) die Summe TP[0]⊕TP[1];
- bei Segment C_{2n-2}: 1) der Null-Wert, 2) das Segment TP[1], 3) die Summe TP[0]⊕TP[1].

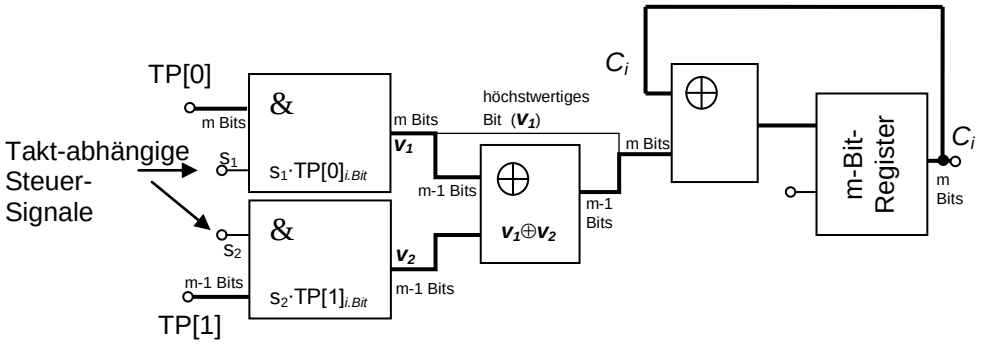


Abbildung 32: Akkumulations-Einheit des Produkt-Segmentes C_i, $1<i<2n-2$

Eine Akkumulation-Einheit mit einer reduzierten Fläche kann für die Produkt-Segmente C_0 und C_{2n-1} entwickelt werden. Hier treten nur 2 verschiedene Fälle bei Produkt-Segmenten auf: in C_0 ist entweder TP[0] oder der Null-Wert je zu akkumulieren, und in C_{2n-1} ist entweder TP[1] oder der Null-Wert zu akkumulieren. **Abbildung 33** zeigt die Struktur solcher Akkumulations-Einheiten am Beispiel des Segmentes C_0.

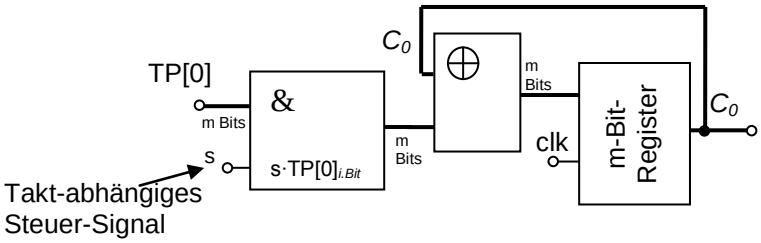


Abbildung 33: Akkumulations-Einheit des Produkt-Segmentes C_0

Die gleiche Idee – also die mehrfache Nutzung der gleichen Einheiten der Schaltung – kann auch für die Ermittlung der TM-Eingänge verwendet werden. **Abbildung 34** zeigt die Struktur der Einheit, bei der XOR-Gatter für die Ermittlung der notwendigen Summen (XOR) der Segmente des Polynoms A(x) sequenziell

mehrfach verwendet werden können. Eine weitere solche Einheit wird für die Ermittlung der Segment-Summen des Polynoms B(x) benötigt.

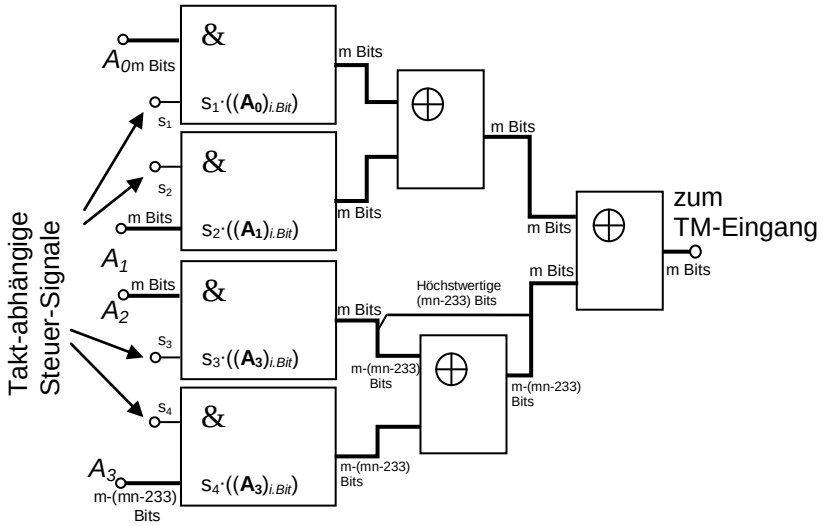


Abbildung 34: Struktur der Einheit für die Ermittlung des TM-Einganges

Abbildung 35 zeigt die Struktur eines seriellen Multiplizierers.

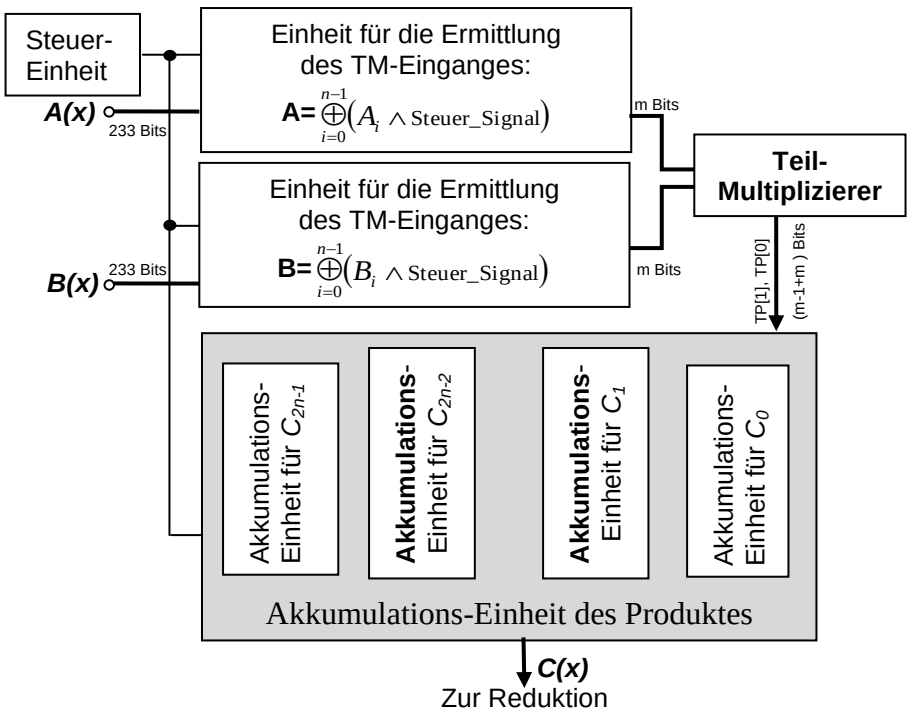


Abbildung 35: Struktur des seriellen Teil-Multiplizierers

Eine weitere Verbesserung der Chip-Parameter des Multiplizierers kann erreicht werden, wenn die Reduktion des Polynom-Produktes in die Akkumulations-Einheit eingebaut wird [28]. In diesem Fall werden nur 233 an Stelle von 465 Flipflops für die Ausgangsregister benötigt. Diese Art der Optimierung ist nicht von der Gatter-Komplexität der MM sondern von der Länge der Multiplikanden abhängig. Deswegen wird sie beim Vergleich der MM nicht berücksichtigt.
Da die optimalen kombinatorischen Teil-Multiplizierer für alle Polynom-Längen bis 600 Bits bereits ermittelt wurden (siehe Kapitel 5 und Anhang 3), wird der serielle Multiplizierer einen TM für die exakt notwendige Operanden-Länge beinhalten. Der serielle 233-Bits-Multiplizierer, mit *n=4* Segmenten, wird z.B. jetzt nicht mehr mit einem 64-Bits-TM ausgestattet [27], [28], sondern mit dem optimalen 59-Bits-Teil-Multiplizierer.

Für jede der untersuchten Multiplikations-Methoden außer für die klassische MM sind die Strukturen der Akkumulations-Einheiten für die Produkt-Segmente C_i, mit $1 \le i \le 2n-2$, in **Abbildung 32** und für die Produkt-Segmente C_0 und C_{2n-1} in **Abbildung 33** optimal.

Für die klassische MM kann eine vereinfachte Akkumulations-Einheit für die Produkt-Segmente C_i, $1 \le i \le 2n-2$, implementiert werden. Hier müssen nur die drei Fälle – entweder TP[0] oder TP[1] oder Null-Wert – akkumuliert werden. **Abbildung 36** zeigt die entsprechende Struktur einer solchen Einheit.

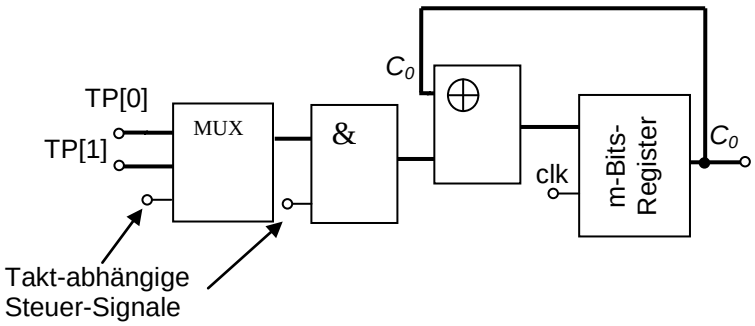


Abbildung 36: Akkumulations-Einheit des Produkt-Segmentes *Ci, 1<i<2n-2* für seriellen Multiplizierer nach der klassischen MM

Nun kann die Gatter-Komplexität des seriellen Multiplizierers entsprechend seiner Struktur (**Abbildung 35)** ermittelt werden. Formel **(170)** zeigt seine Gatter-Komplexität, die aus der GC der einzelnen Einheiten besteht:

$$GC_{nm} = \underbrace{GC_{\text{Steuer-Einheit}}}_{} + 2 \cdot \underbrace{GC_{\text{Einheit für die Ermittlung TM-Eingänge}}}_{} + GC_{m-TM} + \underbrace{GC_{\text{Akkumulations-Einheit des Produktes}}}_{=GC_{\text{Akkumulations-Einheit für } C_0} + \sum_{i=1}^{2n-2} GC_{\text{Akkumulations-Einheit für } C_i} + GC_{\text{Akkumulations-Einheit für } C_{2n-1}}}$$

(170)

Die Steuer-Einheit hat im Vergleich zu den anderen Einheiten des Multiplizierers eine eher geringe Fläche und einen geringen Energieverbrauch. So kann die GC aus **(170)** folgendermaßen abgeschätzt werden:

$$GC_{nm} > 2 \cdot \underbrace{\left((233)_{AND} + ((n-2)m + (233-(n-1)m))_{XOR}\right)}_{GC_{\text{Einheit für die Ermittlung TM-Eingänge}}} + GC_{m-TM} +$$

$$+ \underbrace{m_{AND} + m_{XOR}}_{=GC_{\text{Akkumulations-Einheit für } C_0}} + \underbrace{(2n-2) \cdot ((2m-1)_{AND} + (2m-1)_{XOR})}_{=\sum_{i=1}^{2n-2} GC_{\text{Akkumulations-Einheit für } C_i}} + \underbrace{(m-1)_{AND} + (m-1)_{XOR}}_{=GC_{\text{Akkumulations-Einheit für } C_{2n-1}}} =$$

$$= GC_{m-TM} + (466 + (2n-1)(2m-1))_{AND} + (233 + m - 1 + (2n-2)(2m-1))_{XOR} =$$

$$= GC_{m-TM} + (467 + 4nm - 2n - 2m)_{AND} + (234 + 4nm - 2n - 3m)_{XOR}$$

(171)

Die Signalverzögerung des Multiplizierers kann mit Formel **(172)** ermittelt werden.

$$delay_{nm} = \underbrace{1 \cdot T_{AND} + 2 \cdot T_{XOR}}_{\text{Einheit für die Ermittlung TM-Eingänge}} + delay_{m-TM} + \underbrace{1 \cdot T_{AND} + 2 \cdot T_{XOR}}_{\text{Akkumulations-Einheit für Segment } C_i}$$

(172)

Formel **(173)** zeigt die Gatter-Komplexität des seriellen Multiplizierers nach der klassischen MM und Formel **(174)** zeigt seine Signalverzögerung:

$$GC_{nm}^{clas} > 2 \cdot \underbrace{\left(233_{AND} + ((n-2)m + (233-(n-1)m))_{XOR}\right)}_{GC_{\text{Einheit für die Ermittlung TM-Eingänge}}} + GC_{m-TM} +$$

$$+ \underbrace{m_{AND} + m_{XOR}}_{=GC_{\text{Akkumulations-Einheit für } C_0}} + \underbrace{(2n-2) \cdot (m_{AND} + m_{XOR} + m_{MUX})}_{=\sum_{i=1}^{2n-2} GC_{\text{Akkumulations-Einheit für } C_i}} + \underbrace{(m-1)_{AND} + (m-1)_{XOR}}_{=GC_{\text{Akkumulations-Einheit für } C_{2n-1}}} =$$

$$= GC_{m-TM} + (465 + 2nm)_{AND} + (465 + 2nm - 2m)_{XOR} + (2n-2) \cdot m_{MUX}$$

(173)

$$delay_{nm} = \underbrace{1 \cdot T_{AND} + 2 \cdot T_{XOR}}_{\text{Einheit für die Ermittlung TM-Eingänge}} + delay_{m-TM} + \underbrace{1 \cdot T_{MUX} + 1 \cdot T_{AND} + 1 \cdot T_{XOR}}_{\text{Akkumulations-Einheit für Segment } C_i}$$

(174)

Die Formeln **(171) – (174)** und die Gatter-Komplexitäten der in Kapitel 5 erläuterten optimalen kombinatorischen Multiplizierer wurden als Input-Daten für Algorithmus 2 verwendet.

Im nächsten Abschnitt werden die berechneten Flächen und der Energieverbrauch verschiedener serieller Multiplizierer, die in **Algorithmus 2** bewertet wurden, graphisch dargestellt. Die ausgewählten Implementierungs-Kandidaten wurden synthetisiert, um die theoretisch berechneten Chip-Parameter zu evaluieren.

6.2. Evaluierung der Implementierungs-Kandidaten

Die maximal zulässige Ausführungszeit der Multiplikation der Polynome bestimmt die möglichen Segmentierungen der Operanden. Im Rahmen dieser Arbeit wurden die Segmentierungen untersucht, die nicht mehr als 25 Ausführungs-Takte für die Berechnung des Polynom-Produktes benötigen. Insgesamt wurden die folgenden Fälle untersucht:

- Klassische MM für die Segmentierung der Polynome in bis zu 5 Segmente
- Karatsuba-n-Segment MM für die Segmentierung der Polynome in bis zu 7 Segmente
- Winograd-n-Segment MM für die Segmentierung der Polynome in bis zu 7 Segmente
- Generalisierte Karatsuba MM für die Segmentierung der Polynome in bis zu 7 Segmente
- Montgomery MM für die Segmentierung der Polynome in 5, 6 und 7 Segmente

Die Segmentierung der Polynome bestimmt die Länge der Eingänge des Teil-Multiplizierers, der Bestandteil des seriellen Polynom-Multiplizierers ist. Unter Verwendung der gleichen Segmentierung wird unabhängig von der verwendeten MM der gleiche optimale Teil-Multiplizierer ausgewählt.
Die Struktur des seriellen Multiplizierers für verschiedene MM (außer der klassischen MM) ist gleich (siehe **Abbildung 35**). Nur die Steuer-Einheit und die Anzahl der Ausführungs-Takte sind unterschiedlich. Die Steuer-Einheit hat eine sehr geringe Fläche im Vergleich zu dem Rest des Multiplizierers. Deswegen ist zu erwarten, dass die Fläche aller seriellen Multiplizierer (außer dem nach der klassischen MM) ungefähr gleich ist, wenn die gleiche Segmentierung der Operanden verwendet wird. Der Energie-Verbrauch der Multiplizierer, der von der Anzahl der Ausführungs-Takte (d.h. von der Anzahl der Teil-Produkte der verwendeten MM) abhängig ist, wird aber unterschiedlich.

Es ist auch folgendes zu erwarten: in je mehr Segmente die Operanden zerlegt werden, desto kleiner wird die Fläche, aber desto größer wird der Energieverbrauch der Multiplizierer. Um diese Tendenz zu visualisieren, wurden die Chip-Parameter aller berechneten seriellen Multiplizierer verglichen. **Tabelle 47** zeigt diese Chip-Parameter.

Wie erwartet wurde, sind die Flächen aller MM (außer nach der klassischen MM) bei gleicher Segmentierung der Operanden gleich. Nur bei den klassischen n-Segment-Multiplizierern sind die Flächen kleiner. Diese Multiplizierer benötigen aber mehr Ausführungs-Takte. Diese Tatsache verursacht höheren Energieverbrauch der Schaltung und die langsamere Berechnung des Produktes im Vergleich zu den anderen Multiplizierern.

Tabelle 47: Berechnete Chip-Parameter serieller Multiplizierer

MM	Segmentierung n	Fläche, mm^2	Energie, pJ	Anzahl der Takte
klassische	2	0.137957	162.3821	4
	3	0.079099	195.8255	9
	4	0.057381	297.8388	16
	5	0.045217	441.5716	25
Karatsuba-n-Segment	2	0.141357	55.86557	3
	3	0.083578	122.4479	6
	4	0.062432	192.7682	9
	5	0.050506	306.228	14
	6	0.043737	398.5133	18
	7	0.04062	520.7505	23
Winograd-n-Segment	2	0.141357	55.86557	3
	3	0.083578	122.4479	6
	4	0.062432	214.1826	10
	5	0.050506	306.228	14
	6	0.043737	398.5133	18
	7	0.04062	566.0318	25
Generalisierte Karatsuba-n-Segment	2	0.141357	55.86557	3
	3	0.083578	122.4479	6
	4	0.062432	214.1826	10
	5	0.050506	328.0996	15
	6	0.043737	464.929	21
	7	0.04062	633.9538	28
Montgomery	5	0.050506	284.3564	13
	6	0.043737	376.3747	17
	7	0.04062	498.1099	22

In **Abbildung 37** wurden die Chip-Parameter der Multiplizierer als eine Funktion der Anzahl der Ausführungs-Takte dargestellt. Die rote Kurve verbindet die minimalen Energieverbräuche und die grüne Kurve die minimalen Chip-Flächen der Multiplizierer.

In **Abbildung 37** ist zum Vergleich ein in [28] präsentierter 9-Takt-Multiplizierer gezeigt. Er benutzt einen 64-Bits-TM. Formel **(175)** zeigt die Gatter-Komplexität des seriellen Multiplizierers entsprechend [28].

$$GC_{nm} > 2 \cdot \underbrace{\left((mn)_{AND} + ((n-1)m)_{XOR}\right)}_{\substack{\text{Einheit für} \\ \text{die Ermittlung} \\ \text{TM-Eingänge}}} + GC_{m-TM} +$$

$$+ \underbrace{((4mn-2m)_{AND} + (4mn-2m)_{XOR})}_{\substack{\text{Akkumulations-} \\ \text{Einheit}}} = GC_{m-TM} + (6mn-2m)_{AND} + (6mn-4m)_{XOR}$$

(175)

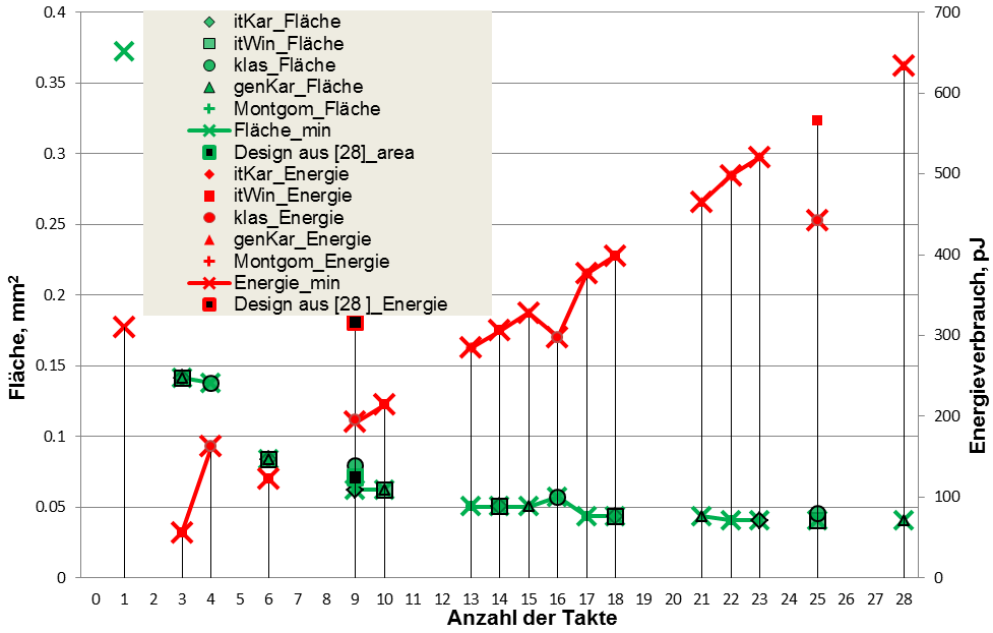


Abbildung 37: Berechnete Chip-Parameter der seriellen Multiplizierer als Funktion der Anzahl der Takte.

Aus dem Vergleich der Chip-Parameter folgt, dass der serielle 3-Takt-Multiplizierer den minimalen Energieverbrauch hat. Bei ihm wurde die Segmentierung der Operanden in 2 Terme entsprechend der Karatsuba-MM verwendet. Dieser Multiplizierer ist der Energie-effizienteste Implementierungs-Kandidat. Zwei weitere serielle Multiplizierer – der 6- und der 9-Takt-Multiplizierer – können auch als Implementierungs-Kandidaten gewählt werden. Im Vergleich zum Energie-effizientesten Multiplizierer haben sie eine zwei Mal kleinere Fläche und somit kleinere Herstellungskosten. Bei allen anderen Mehr-Takt-Multiplizierern führt eine unwesentliche Reduzierung der Chip-Fläche zu deutlich höherem Energie-Verbrauch und einer deutlich größeren Ausführungs-Zeit.

6.3. Synthese-Ergebnisse

Die Evaluierung der theoretisch berechneten Chip-Parameter mit Synthese-Daten wurde für 6- und 9-Takt seriellen Multiplizierer durchgeführt.

Diese Multiplizierer wurden ins IHP ECC-Chip eingebaut. Das alte ECC-Design benutzt einen 9-Takt seriellen Multiplizierer für die EC Punkt Multiplikation *kP*. Die Struktur des Multiplizierers wurde in [28] veröffentlicht. Formel **(**175**)** zeigt seine Gatter-Komplexität. Dieser serielle Multiplizierer beinhaltet einen 64-Bits-Teil-Multiplizierer. Um die Flächen- und die Energie-Reduktion bei Verwendung eines kleineren TM zu evaluieren, wurde der alte 9-Takt-Multiplizierer erst durch einen neuen 9-Takt-Multiplizierer und danach durch einen neuen 6-Takt-Multiplizierer ersetzt. **Tabelle 48** zeigt die Chip-Parameter dieser drei Designs. Die Synthese wurde mit dem Design Vision von Synopsys [54] für die IHP Technologie 0,13µ durchgeführt.

Die Reduktions-Einheit wurde in den seriellen Multiplizierer eingebaut. Die Synthese-Ergebnisse in **Tabelle 48** zeigen die Parameter des Multiplizierers mit der Reduktions-Einheit.

Der neue 9-Takt Multiplizierer beinhaltet einen 59-Bits-Teil-Multiplizierer. Der 6-Takt-Multiplizierer beinhaltet einen 78-Bits-TM. Diese TM wurden mittels **Algorithmus 1** als die Flächen-optimalen Kombinationen der MM aus der folgenden MM gewählt. Zu dieser Menge der MM gehören die folgenden MM:

- Klassische MM
- Iterativ optimierte generalisierte Karatsuba-MM
- Iterativ optimierte Karatsuba-n-Segment-MM für die Segmentierung der Polynome in bis zu 16 Segmente
- Iterativ optimierte Winograd-n-Segment-MM für die Segmentierung der Polynome in bis zu 16 Segmente

Tabelle 49 zeigt die Flächen-optimalen Kombinationen der MM aus dieser Menge für 59- und 78-Bits-TM für die IHP-Technologie 0,13µ. Der 64-Bits-TM wurde entsprechend [28] synthetisiert. Seine Parameter sind auch in **Tabelle 49** zum Vergleich gegeben.

Tabelle 48: kP-Designs mit verschiedenen seriellen Multiplizierern für EC B-233 [8]

Chip-Parameter		kP-Design für IHP 0,13µ		
		Design aus [28]	mit neuem 9-Takt Multiplizierer	mit neuem 6-Takt Multiplizierer
TM	Eingangs-Länge	64-Bits Operanden	59-Bits Operanden	78-Bits Operanden
	Fläche, mm^2	0.039585	0.036177	0.057458
	Signal-Verzögerung, ns	2,41	2,22	2,50
	Energie=power·T, pJ	$9{,}94\cdot10^{-4}$W·30ns=29,82	$9{,}04\cdot10^{-4}$W·30ns=27,12	$14{,}3\cdot10^{-4}$W·30ns=42,9
Serieller Multiplizierer	Eingangs-Länge	233-Bits Operanden	233-Bits Operanden	233-Bits Operanden
	Reduktions-Einheit	für EC B-233	für EC B-233	für EC B-233
	Anzahl der Ausführungs-Takte	9	9	6
	Fläche, mm^2	0,113992	0,108979	0,132332
	Signal-Verzögerung (ohne TM), ns	0,28	0,29	0,26
	Energie pro Takt, pJ	116,7	105	120
	Energie pro Polynom-Multiplikation, pJ	1050.3	945	720
kP-Design	Anzahl der Ausführungs-Takte	13614	13614	9861
	Ausführungs-Zeit, ms	0,41	0,41	0,30
	Fläche, mm^2	0,24796	0,24295	0,26621
	Signal-Verzögerung, ns	25,41	25,88	25,71
	Maximale Taktrate, MHz	39,4	38,6	38,9
	Arbeits-Frequence, MHz	33	33	33
	Energie pro Takt, pJ	168,3	157,5	176,4
	Energie pro kP, pJ	2 291 236	2 144 205	1 739 480

Tabelle 49: Chip-Parameter der synthetisierten TM

n	n'	optimale Kombination der MM	Theorie				Synthese	
			GC	delay, ns	area, mm^2	energy, pJ	area, mm^2	delay, ns
64	64=	4(itK)·4(itW)·4(klas)	$(1296_{AND}, 2387_{XOR})$	$1\cdot T_{\&}+12\cdot T_{XOR}=1.14$	0.042563	35.1340	0.039585	2,41
59	60=	2(itK)·6(itW)·5(klas)	$(1350_{AND}, 2094_{XOR})$	$1\cdot T_{\&}+13\cdot T_{XOR}=1.22$	0.039057	32.7636	0.036177	2,22
78	80=	4(itK)·4(itK)· 5(klas)	$(2025_{AND}, 3396_{XOR})$	$1\cdot T_{\&}+14\cdot T_{XOR}=1.31$	0.062016	51.6368	0.057458	2,50

Im Vergleich zu den optimalen Kombinationen der 10 MM (siehe Kapitel 5) ist die GC der 59-Bits-TM aus **Tabelle 49** um 5 AND- und 23 XOR-Gatter größer. Die GC des 78-Bits-TM ist um 69 AND- und 30 XOR-Gatter größer. Die ermittelten GC für Polynom-Längen bis 600 Bits sind im Anhang 4 gegeben. Dem GC-Unterschied des 59-Bits-TMs entspricht eine Flächen-Differenz von unter 0,1% der gesamten Fläche des ECC-Designs bzw. 0,3% der Fläche des ECC-Designs bei dem 78-Bits-TM. Die Implementierung der neuen Struktur des seriellen Multiplizierers mit ihrer GC nach **(171)** kann die gesamte Fläche des ECC-Designs mit dem 9-Takt-Multiplizierer nur um 1% im Vergleich zur alten Struktur mit der GC nach **(175)** reduzieren. Beim ECC-Design mit dem 6-Takt-Multiplizierer beträgt der Unterschied nur 0,8%. Wegen der geringen Flächen-Reduktion des Chips – etwa um 0,002 mm^2 – wurden im Rahmen dieser Arbeit die neue Struktur des seriellen Multiplizierers und die optimalen Kombinationen aus 10 MM für die 59- und 78-Bits-TM nicht implementiert.

Abbildung 38 stellt die Daten aus **Tabelle 48** graphisch dar, d.h. sie zeigt die Fläche und den Energieverbrauch der synthetisierten Designs: 59-, 64- und 78-Bits-TM; drei serielle Multiplizierer, je mit einem dieser 3 TM; und 3 kP-Designs, je mit einem der 3 seriellen Polynom-Multiplizierer.

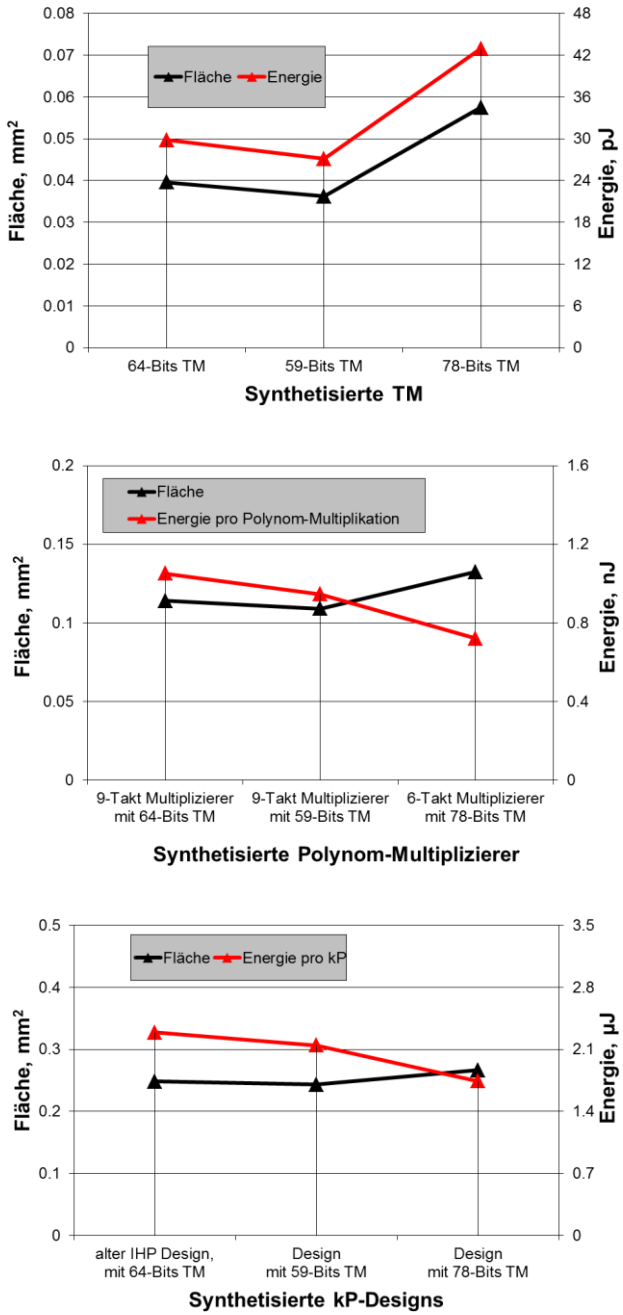


Abbildung 38: Chip-Parameter der synthetisierten Designs.

Aus dem Vergleich der Chip-Parameter der synthetisierten ECC-Designs folgt:

- Der Unterschied bei der maximalen Taktrate aller drei Designs beträgt weniger als 2%.
- Die Verwendung des optimalen 59-Bits-TM anstatt des 64-Bits-TM in dem 9-Takt-Polynom-Multiplizierer reduziert die Chip-Fläche des kP-Designs um 2% und den Energieverbrauch um 6,4%.
- Die Verwendung des 6-Takt Polynom-Multiplizierers mit dem optimalen 78-Bits-TM erhöht die Chip-Fläche des kP-Designs um 7%, was der absoluten Flächen-Differenz von nur 0,018 mm^2 entspricht. Der Energieverbrauch und die Ausführungs-Zeit der kP-Operation wurden aber um 24% bzw. 28% reduziert.

Die Chip-Parameter der synthetisierten Designs bestätigen die theoretischen Berechnungen. Die Abweichung zwischen berechneten und synthetisierten Werten beträgt 7% bei den TM-Flächen und 17% bei den TM-Energien. Die Ursachen für diese Abweichungen wurden bereits im Kapitel 5.2.3 erklärt.

Kapitel 7

Zusammenfassung und Ausblick

In diesem Kapitel werden die Ergebnisse dieser Arbeit zusammengefasst und die weiteren Forschungs-Richtungen kurz dargestellt.

7.1. Zusammenfassung

Die Anwendung asymmetrischer Kryptosysteme, z.B. ECC, erfordert große Rechenkapazität von mobilen Geräten oder drahtlosen Sensorknoten. Die Implementierung der ECC in Hardware reduziert den Zeit- und Energie- Aufwand. Aufgrund der Herstellungskosten und des Energieverbrauches, ist die optimale Hardware-Implementierung sehr wichtig. Die Multiplikation der $GF(2^n)$-Elemente ist eine aufwendige und oft aufgerufene Operation der ECC. Diese Arbeit hat sich auf die flächen- und/oder energieeffiziente Hardware-Implementierung der Polynom-Multiplikation, die ein Schritt der Multiplikation der $GF(2^n)$-Elemente ist, konzentriert.
Flächen- und/oder energieeffiziente Polynom-Multiplizierer für *n*-Bits große Operanden wurden im Rahmen dieser Arbeit als serielle Multiplizierer auf Basis Karatsuba-ähnlicher MM entwickelt. Die Wahl des effizientesten Implementierungs-Kandidaten wurde am Beispiel des Multiplizierers für 233-Bits langen Operanden gezeigt. Diese Operanden-Länge – 233-Bits – gilt als sichere Schlüssellänge für die Periode 2011 – 2030 [8].
Viele Multiplikations-Methoden für die Multiplikation großer ganzer Binär-Zahlen können für die Polynom-Multiplikation angepasst werden. Bei den Multiplikations-Methoden werden die Operanden segmentiert. Das Polynom-Produkt wird als eine Summe der Segment-Produkte (oder Teil-Produkte) berechnet. Jede Multiplikations-Methode hat eine eigene Anzahl der Teil-Produkte und damit gebundene Komplexität, die als die Anzahl an AND- und XOR-Gattern für die Hardware-Implementierung dargestellt werden kann. Bei seriellen (Mehr-Takt-)

Multiplizierer können alle Teil-Produkte seriell – taktweise – von nur einem Teil-Multiplizierer berechnet werden, was die Fläche des Multiplizierers wesentlich reduziert. Die verwendete Multiplikations-Methode bestimmt dann die Anzahl der Ausführungs-Takte. Die Chip-Parameter der Multiplizierer können aus den Komplexitäten der verwendeten Multiplikations-Methoden ermittelt und verglichen werden, um den Implementierungs-Kandidaten zu bestimmen.
Karatsuba-ähnliche MM haben eine kleinere Anzahl an Teil-Produkte als die klassische MM, benötigen aber eine größere Anzahl an Additionen. Deswegen ist der Anteil an XOR-Gatter bei der Implementierung Karatsuba-ähnlicher MM immer größer als bei der klassischen MM. Hinzu kommt, dass die Parameter der XOR-Gatter größer sind als die der AND-Gatter. Bei der Multiplikation der Operanden kleiner Länge ist die gesamte Anzahl der Gatter bei Karatsuba-ähnlichen MM vergleichbar oder größer als bei der klassischen MM. Aus diesem Grund ist die rekursive Anwendung Karatsuba-ähnlicher Multiplikations-Methoden nur bis zu einer gewissen Länge der Segmente sinnvoll. Die Segmente kleiner Länge werden nach der klassischen MM multipliziert. Ein flächen- oder/und energie-optimaler Polynom-Multiplizierer kann als Kombination mehrerer Karatsuba-ähnlicher Multiplikations-Methoden mit der klassischen Multiplikations-Methode realisiert werden.
Da die Parameter der XOR-Gatter größer sind als die der AND-Gatter, können die Chip-Parameter des Polynom-Multiplizierers wesentlich verbessert werden, wenn die Anzahl der XOR-Gatter bei gleicher Anzahl AND-Gatter reduziert wird. Diese Art der Optimierung kann bei vielen Karatsuba-ähnlichen Multiplikations-Methoden durchgeführt werden. Die Reduktion der Anzahl der XOR-Gatter kann durch Optimierung der Reihenfolge der Additionen der Segment-Produkte erreicht werden. Eine optimale Kombination auf diese Weise optimierten Multiplikations-Methoden, kann die Fläche und den Energieverbrauch des Polynom-Multiplizierers wesentlich reduzieren.
Bei der Hardware-Implementierung eines Algorithmus muss auch die Tatsache berücksichtigt werden, dass die Hersteller-Technologie das Flächen- und das Energie-Verhältnis eines XOR- und eines AND-Gatter bestimmt. Deswegen kann die flächen- oder/und energie-optimale Kombination der Multiplikations-Methoden für jede Technologie eine andere sein.

Im Rahmen dieser Arbeit wurden folgenden Methoden entwickelt und verwendet:

1 – Bestimmung der Berechnungs-Reihenfolge der Additionen der Teil-Produkte, die eine minimierte Anzahl an XOR-Gattern für die Berechnung des Polynom-Produktes benötigt.

> 10 Multiplikations-Methoden wurden untersucht. Für 9 untersuchten MM (außer der klassischen MM) wurden eine optimierte Reihen-Folge der

Berechnung und ihre Gatter-Komplexität ermittelt. Die klassische MM kann auf diese Weise nicht optimiert werden. Der Einsatz dieser Optimierung kann die Komplexität der MM wesentlich reduzieren. Zum Beispiel bei der generalisierten Karatsuba MM [18] beträgt die Reduktion des XOR-Aufwandes durchschnittlich 39 % für die Polynom-Längen bis 600 Bits. Für die IHP 0,13µ-Technologie entspricht diese Reduktion des XOR-Aufwandes einer durchschnittlichen Flächen-Reduktion der Polynom-Multiplizierer um 35 %. Bei der 4-Segment-Karatsuba-MM wird nicht nur der XOR-Aufwand reduziert, sondern auch die Signal-Verzögerung im Vergleich zur rekursiven Anwendung der originalen Karatsuba-MM.

2 – Die algorithmische Bestimmung der flächen- oder/und energie-optimalen Kombination der untersuchten Multiplikations-Methoden.

Der vorgeschlagene Algorithmus ist technologieunabhängig und kann für die Suche der flächen- oder/und energie-optimalen Kombination der MM für beliebig große Polynome verwendet werden. Es wurden die optimalen Kombinationen der 10 MM gesucht: der klassischen MM und der 9 iterativ optimierten MM (d.h. der MM, für die eine Reihen-Folge der Operationen mit reduziertem XOR-Aufwand im Rahmen dieser Arbeit bestimmt wurde). Mittels des vorgeschlagenen Algorithmus wurden die flächen- und die energie-optimalen Kombinationen der MM für die Polynom-Längen bis zu 600 Bits bestimmt. Alle ECC-relevanten Polynom-Längen liegen in diesem Bereich. Die durchschnittliche Reduktion der Flächen-Werte im Vergleich zu den rekonstruierten Daten aus [30] beträgt 12 %. Für alle ECC-relevanten Polynom-Längen wurde der Einfluss der Hersteller-Technologie auf die Ergebnisse des Algorithmus untersucht. Die Flächen-optimalen Kombinationen der MM für 283- und für 571-Bits Polynome sind auch Energieverbrauchs-optimal und von der Hersteller-Technologie unabhängig. Die theoretisch berechneten Chip-Parameter der Polynom-Multiplizierer für alle ECC-relevanten Polynom-Längen wurden mit Synthese-Daten erfolgreich evaluiert.

3 – Implementierung eines seriellen Mehr-Takt-Multiplizierers für 233-Bits Polynome auf Basis Karatsuba-ähnlicher Multiplikations-Methoden.

Für die Bestimmung des Implementierungs-Kandidaten wurden die klassische MM und 4 Karatsuba-ähnliche MM unter Verwendung der Segmentierung der Operanden in bis zur 7 Segmente untersucht. Jeder Multiplizierer beinhaltete nur einen Teil-Multiplizierer. Für alle Teil-Multiplizierer wurden die flächen-optimalen Kombinationen der 10 MM algorithmisch bestimmt. Der Vergleich der theoretisch berechneten Chip-

Parameter zeigt, dass der serielle 3-Takt-Multiplizierer unter Verwendung der Segmentierung der Operanden in 2 Terme entsprechend der Karatsuba-MM der energieeffizienteste Polynom-Multiplizierer ist. Als die Implementierungs-Kandidaten wurden aber der 6-Takt-Multiplizierer (unter der Verwendung der Segmentierung der Operanden in 3 Terme entsprechend der Winograd-MM) und der 9-Takt-Multiplizierer (unter der Verwendung der Segmentierung der Operanden in 4 Terme entsprechend der Karatsuba-MM) bestimmt. Im Vergleich zum energieeffizientesten Multiplizierer haben sie eine wesentlich kleinere Fläche, und somit die kleineren Herstellungskosten. Bei allen anderen Mehr-Takt-Multiplizierern führt eine unwesentliche Reduzierung der Chip-Fläche zu deutlich höherem Energie-Verbrauch und einer deutlich größeren Ausführungs-Zeit.

Die Chip-Parameter der synthetisierten Designs bestätigen erfolgreich die theoretischen Berechnungen. Die Verwendung des seriellen 6-Takt-Polynom-Multiplizierer mit dem Flächen-optimierten 78-Bits Teil-Multiplizierer reduziert den Energieverbrauch und die Ausführungszeit des Designs für die kP-Operation um 24 % bzw. 28 %. Die Fläche des ganzen kP-Designs wird nur um 0,018 mm^2 größer als die Fläche des nach[28] implementierten kP-Design.

7.2. Ausblick

Auf Basis der Ergebnisse dieser Arbeit können noch folgende Aspekte der Optimierungen untersucht werden:

1- Berücksichtigung der Signal-Verzögerung der Schaltung bei der algorithmischen Bestimmung der optimalen Kombination der MM (siehe Algorithmus 1, Kapitel 3.5).

Jede in Hardware implementierte MM hat eine eigene Anzahl an AND- und XOR-Gattern und ihre spezifische Signalverzögerung. Die kleinste Signal-Verzögerung verursacht die klassische MM, die für die Operanden kleiner Länge (1- bis 5- und 7-Bits Operanden) auch Flächen-optimal ist. Die iterative Berechnung der Produkt-Segmente verursacht eine größere Signal-Verzögerung als die separate Berechnung. Wenn die Takt-Frequenz ein kritischer Parameter der Schaltung ist, kann eine flächen- und signalverzögerungs-optimale Kombination der MM algorithmisch bestimmt werden. Dies kann z. B. mit der Einführung der folgenden Bedingung erreicht werden: wenn die Fläche einer der untersuchten Kombinationen der MM nicht mehr als a % größer ist, und ihre Signal-Verzögerung mehr als b % kleiner ist, kann diese Kombination als die

optimale ausgewählt werden. In diesem Fall wäre es sinnvoll auch die MM mit separater Berechnung der Produkt-Segmente in die Menge der zu untersuchenden MM aufzunehmen.
Eine andere Möglichkeit ist es die klassische MM als die optimale für alle Multiplizierer kleiner Länge, z. B. bis zu 12-Bits langen Operanden, zu bestimmen. Die Änderungen in Algorithmus 1, die diese Initialisierung implementieren, sind minimal und basieren auf der Tatsache, dass die klassische MM für n-Bits Operanden mit $n \in \{6, 8, 9, 10, 11, 12\}$ durchschnittlich um 8 % größer ist, aber eine 40 % kleinere Signalverzögerung verursacht.

2- In [55] wird der Einsatz von Toeplitz-Matrizen zur Reduktion der Komplexität der Multiplikation von *GF(2^n)*-Elementen vorgeschlagen. Eine praktische Bedeutung hat der Vergleich dieser MM mit Standard-Lösung, die aus der flächen-optimalen Polynom-Multiplizierer und aus der Reduktions-Einheit besteht.

Normalerweise bestehen die *GF(2^n)*-Multiplizierer aus zwei Komponenten: aus dem Polynom-Multiplizierer und aus der Reduktions-Einheit. Die Ergebnisse dieser Arbeit können bei der Implementierung eines universalen *GF(2^n)*-Multiplizierers verwendet werden. Ein Polynom-Multiplizierer für alle standardisierten EC kann beispielweise aus nur einem 571-Bits-Polynom-Multiplizierer und vielen Reduktions-Einheiten bestehen, weil ein Multiplizierer für 571-Bits große Operanden immer auch das Produkt der 163-, 233-, 283- und 409-Bits großen Operanden berechnen kann.
Das Produkt der *GF(2^n)*-Elemente kann auch mittels Toeplitz-Matrizen berechnet werden [55]. Da diese Matrizen über eine Art der Symmetrie verfügen, können sie mit einer reduzierten Anzahl an Operationen multipliziert werden. Diese MM ist für die Implementierung eines universalen *GF(2^n)*-Multiplizierer nicht geeignet, aber sie kann für die Implementierung des flächen-optimierten Multiplizierers für einen ausgewählten *GF(2^n)* verwendet werden. Die Evaluierung der Synthese-Ergebnisse dieser MM mit den Ergebnissen dieser Arbeit hat eine praktische Bedeutung.

3- Entwicklung eines universalen *GF(p)* und *GF(2^n)*-Multiplizierer auf Basis in Rahmen dieser Arbeit durchgeführten Optimierungen.

Auf Basis des flächen-optimalen Polynom-Multiplizierers kann ein universaler Multiplizierer entwickelt werden, der *n*-Bits große Binäre Operanden abhängig vom Steuer-Signal entweder als *n*-Bits Polynome oder als ganze Zahlen bearbeiten kann [56]. Dieser universale

Multiplizierer kann für die Implementierung eines Multiplizierers benutzt werden, der fähig ist, abhängig vom Steuer-Signal entweder die Elemente aus den Galoiskörper *GF(p)* oder die Elemente aus den Galoiskörper $GF(2^n)$ zu berechnen.

Literaturverzeichnis

[1] Rivest, Shamir, Adleman: *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Communications of the ACM, Vol. 21, Nr. 2, 1978, pp. 120–126

[2] N. Koblitz, *Elliptic curve cryptosystems*, in *Mathematics of Computation* Vol. 48, Nr. 177, 1987, pp. 203–209

[3] V. Miller: *Use of elliptic curves in cryptography.* Proceeding CRYPTO ’85 in Advances in Cryptology, Springer-Verlag London, 1986, pp. 417-426

[4] Advanced Encryption Standard - http://csrc.nist.gov/archive/aes/index.html

[5] National Institute of Standards and Technology (NIST) - http://csrc.nist.gov

[6] Bruce Schneier: *Applied Cryptography. Protocols, Algorithms, and Source Code in C*, Second Edition. 1996, ISBN 0-471-11709-9

[7] Peter Langendoerfer, Zoya Dyka, Oliver Maye, Rolf Kraemer: *A Low Power Security Architecture for Mobile Commerce*, 5th IEEE CAS Workshop on Wireless Communications and Networking, IEEE Society Press 2002

[8] *NIST Digital Signature Standard (DSS)*, FIPS PUB 186-3, Juni 2009, http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf

[9] Standards for Efficient Cryptography Group (SECG): *SEC 2: Recommended Elliptic Curve Domain Parameters,* http://www.secg.org/download/aid-784/sec2-v2.pdf

[10] W. Diffie and M. E. Hellman: *New Directions in Cryptography*, IEEE Transactions on Information Theory, Vol. IT-22, Nr. 6, 1976, pp: 644–654

[11] Elaine Barker, Don Johnson, and Miles Smid: *NIST Special Publication 800-56A, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography (Revised),* March, 2007 - http://csrc.nist.gov/publications/nistpubs/800-56A/SP800-56A_Revision1_Mar08-2007.pdf

[12] Standards for Efficient Cryptography Group (SECG): *SEC 1: Elliptic Curve Cryptography* http://www.secg.org/download/aid-780/sec1-v2.pdf

[13] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid: *NIST Special Publication 800-57: Recommendation for Key Management, Part 1: General (Revised)*, March 2007, http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf

[14] Donald E. Knuth: *The Art of Computer Programming*, Vol. 2: *Seminumerical Algorithms*, Addison-Wesley 1998, ISBN 0-201-89684-2

[15] A. Karatsuba, A., Ofman, Y.: *Multiplication of Many-Digital Numbers by Automatic Computers*. Doklady Akad. Nauk SSSR, Vol. 145 (1962), pp: 293–294. Translation in Physics-Doklady, 7 (1963), pp: 595–596

[16] Winograd, S.: *Arithmetic Complexity of Computations*. SIAM, 1980

[17] Sunar, B.: *A Generalized Method for Constructing Subquadratic Complexity GF(2^k) Multipliers*. IEEE Transactions on Computers, Vol. 53, Nr. 9, 2004, pp: 1097-1105

[18] Weimerskirch, A., Paar, C.: *Generalizations of the Karatsuba Algorithm for Efficient Implementations*. Report 2006/224, Cryptology ePrint Archive, 2006, http://eprint.iacr.org/2006/224.pdf

[19] Montgomery, P.L.: *Five, Six, and Seven-Term Karatsuba-Like Formulae. IEEE Transactions on Computers*, Vol. 54, Nr. 3, 2005, pp: 362-369

[20] Haining Fan and M. Anwar Hasan. *Comments on "Five, six, and seven- term Karatsuba-like formulae"*. IEEE Trans. Comput., 56 (5), 2007, pp:716–717

[21] Cenk, M., Koc, C. K., Ozbudak, F.: *Polynomial multiplication over finite fields using field extensions and interpolation*, in 19th IEEE Symposium on Computer Arithmetic, IEEE Computer Society Press, Portland, Oregon, 2009, pp: 84-91.

[22] Oseledets, I.: *Improved n-term Karatsuba-like Formulae in GF(2)*. IEEE Transactions on Computers, Vol. 60, Nr. 8, 2011, pp: 1212-1216

[23] J. M. Pollard. *The Fast Fourier Transform in a Finite Field*, Mathematics of Computation, Vol. 25, 1971, pp: 365-374

[24] S. Yazaki and K. Abe, *VLSI Implementation of Karatsuba Algorithm and Its Evaluation*, Proceeding The International Workshop on Modern Science and Technology 2006, Wuhan, China, May 2006, pp: 378-383

[25] S. Baktir and B. Sunar: *Achieving efficient polynomial multiplication in Fermat fields using the Fast Fourier Transform*, ACM Southeast Regional Conference Proceedings of the 44-th annual Southeast regional conference, ACM Press, 2006, pp: 549-554.

[26] Rodriguez-Henriquez, F., Koc, C.K.: *On fully parallel Karatsuba multipliers for GF(2^m)*. In Proceedings of the International Conference on Computer Science and Technology - CST 2003, Acta Press, Cancun, Mexico (2003), pp: 405-410.

[27] Dyka, Z., Langendoerfer, P.: *Area efficient hardware implementation of elliptic curve cryptography by iteratively applying Karatsubas method*, In Proceedings of the Design, Automation and Test in Europe (DATE 2005), 2005, Vol.3, pp: 70-75

[28] Peter, S., Langendoerfer, P.: *An Efficient Polynomial Multiplier GF(2^m) and its Application to ECC Designs*. In Proceedings of the Design, Automation and Test in Europe (DATE 2007), 2007, pp:1253-1258

[29] J. von zur Gathen and J. Shokrollahi: *Efficient FPGA-based Karatsuba multipliers for polynomials over F_2*, in Selected Areas in Cryptography (SAC) 12th International Workshop, Canada, 2005, Revised Selected Papers, Vol. 3897 of LNCS, Springer-Verlag, 2006, pp: 359–369

[30] J. von zur Gathen and J. Shokrollahi: Fast arithmetic for polynomials over F2 in hardware. In IEEE Information Theory Workshop (2006), pp: 107–111. IEEE, Punta del Este, Uruguay

[31] Innovations for High Performance Microelectronics, http://www.ihp-microelectronics.com/

[32] Daniel V.Bailey, C. Paar: *Optimal extension fields for fast arithmetic in public-key algorithms*. In: Advances in Cryptology-CRYPTO '98, vol. 1462 of LNCS, pp: 472-485. Springer Verlag, 1998

[33] Darrel Hankerson, Alfred Menezes, *Scott Vanstone: Guide to Elliptic Curve Cryptography*, Springer-Verlag New York, Inc., 2004, ISBN 0-387-95273-X

[34] Julio López Hernandez, Ricardo Dahab: *Improved Algorithms for Elliptic Curve Arithmetic in GF(2^n)*, SAC 1998, LNCS 1556, Springer-Verlag, 1998, pp: 201-212

[35] Montgomery, P. L.: *Speeding the Pollard and Elliptic Curve Methods of Factorization, Mathematics of Computation*, Vol.48, 1987, pp: 243-264

[36] López, J. and Dahab, R.: *Fast Multiplication on Elliptic Curves over GF(2^m) without Precomputation*, In Proceedings of CHES. 1999, LNCS 1717, Springer-Verlag, 1999, pp: 316-327

[37] R. Schroeppel, H. Orman, and S. O'Malley: *Fast key exchange with elliptic curve systems*, Tech. Rep. 95-03, Department of Computer Science, The University of Arizona, Tucson, Ariz, USA, March 1995

[38] D. Hankerson, JL Hernandez and A. Menezes: *Software implementation of elliptic curve cryptography over binary fields*, CHES 2000, LNCS 1965, Springer- Verlag, pp: 1-24, 2000

[39] S. C. Shantz: *From Euclid's GCD to Montgomery multiplication to the great divide*, Tech. Rep. TR-2001-95, Sun Microsystems Laboratories, Santa Clara, Calif., USA, June 2001.

[40] H. Eberle, N. Gura, S. C. Shantz and V. Gupta: *A cryptographic Processor for arbitrary Elliptic Curve over GF(2^m)*, Tech. Rep. TR-2003-123, Sun Microsystems Laboratories, Santa Clara, Calif., USA, Mai 2003.

[41] Nils Gura, Hans Eberle: *Generic Implementations of Elliptic Curve Cryptography using Partial Reduction*. CCS 2002, 9th ACM Conference on Computer and Communications Security, Washington, DC, November 17-21, 2002, pp: 108-116.

[42] Nils Gura, Hans Eberle, Arun Patel: *Comparing ECC and RSA on Small Devices*, CHES 2004, Workshop on Cryptographic Hardware and Embedded Systems, Cambridge (Boston), USA, 2004, pp: 11-13

[43] Bailey, D. V. and Paar, C.: *Efficient Arithmetic in Finite Field Extensions with Application in Elliptic Curve Cryptography*. Journal of Cryptology, Vol. 14, Nr. 3, 2001, pp:153-176

[44] Blahut, R.E.: *Fast Algorithms for Digital Signal Processing*, Addison-Wesley Publishing Company Reading, Massachussetts, 1985, reprinted by Cambridge University Press, 2010, ISBN: 0521190495, 0511776373

[45] Horowitz, PI., Hill, W.: *The Art of electronics*, Cambridge University Press, New York (1989)

[46] C. Grabbe,M. Bednara, M. Daldrup, J. Teich, J. von zur Gathen, J. Shokrollahi: FPGA Designs of parallel high performance GF(2^{233}) Multipliers, In Proc. of the IEEE International Symposiumon Circuits and Systems (ISCAS-03), Vol 2, Bangkok, Thailand, pp: 268-271

[47] Haining Fan, Jiaguang Sun, Ming Gu and Kwok-Yan Lam: *Overlap-free Karatsuba-Ofman polynomial multiplication algorithm,*. IET Information security, Vol. 4, Nr.1, 2010, pp: 8-14

[48] P. Langendoerfer, Z.Dyka, S. Peter: *Method and apparatus for calculating a polynomial multiplication, in particular for elliptic curve cryptography*, US Patent Application Publication, Pub. No. US 2009/0136022, Pub. Date: 2009

[49] Gang Zhou, Harald Michalik, and László Hinsenkamp: *Complexity Analysis and Efficient Implementations of Bit Parallel Finite Field Multipliers Based on Karatsuba-Ofman Algorithm on FPGAs*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 18, Nr. 7, 2010, pp: 1057-1066

[50] M. Machhout, M. Zeghid, W. Elhadjyoussef, B. Bouallegue, A. Baganne, and R. Tourki: *Efficient Large Numbers Karatsuba-Ofman Multiplier Designs for Embedded Systems*, International Journal of Electrical and Computer Engineering Vol.4, Nr. 9, 2009, pp: 548-557

[51] K.R. Reischuk, Komplexitätstheorie, Band I: Grundlagen: Maschinenmodelle, Zeit- und Platzkomplexität Teubner Verlag Stuttgart, 1999, ISBN 3-519-12275-8

[52] Landau, E.: Handbuch der Lehre von der Verteilung der Primzahlen, Leipzig, Germany: Teubner, 1909. Reprinted by New York: Chelsea, 1953

[53] David L. Applegate, Robert E. Bixby, Vasek Chvátal & William J. Cook: The Traveling Salesman Problem: A Computational Study, Princeton University Press, 2006

[54] Synopsis, http://www.synopsys.com/

[55] Haining Fan and M. Anwar Hasan: *A New Approach to Subquadratic Space Complexity Parallel Multipliers for Extended Binary Fields*, IEEE Transaction on computers, Vol. 56, Nr. 2, 2007, pp: 224-233

[56] Z.Dyka, P. Langendoerfer: *Unifizierter Multiplizierer für die Galois-Körper GF(2^n) und GF(p), sowie Kryptographie-Verfahren und Kryptographie-Vorrichtung*, Anmeldung zur Patent von 16.11.2010, Anmeldeaktenzeichen: 10 2010 043 993.2

Anhänge

Anhang 1

Berechnung der Karatsuba-*n*-GR

Im Kapitel 4.3.4.2 wurden Polynome der Länge *n* einer der vier GR-Gruppen zugeordnet. Die Berechnung der n-GR aus der GR-Gruppe 1 wurde bereits im Kapitel 4.3.4.2 vollständig erklärt. Auch die Berechnung der rechten *n-GR*-Seiten wurde im gleichen Kapitel erklärt. Hier werden die Sonderfälle der Berechnung der linken Seiten anderer n-GR-Gruppen untersucht. Außerdem werden hier die weiteren, für den Algorithmus 3 (siehe Kapitel 4.3.3) notwendigen Daten, ermittelt: der XOR-Aufwand der *voll iterativen* Berechnung der linken (*n-1*) Spalten-Werte der *n*-GR so wie auch der XOR-Aufwand der Berechnung der mittleren n-GR-Spalte $c^{n\text{-}GR}{}_{n\text{-}1}$ nach *voll iterativem* und nach optimalem Ablaufplan der Berechnung der linken *n-GR*-Seite. Dieselben Daten werden auch für die *nicht vollen n*-GR hergeleitet.
In diesem Anhang werden einige Bilder, Tabellen und Formeln aus Kapitel 4 sowie auch die Merkmale der n-GR-Gruppen wiederholt angegeben, um den Text leichter lesbar zu machen. Anhang 1 bis Anhang 4 haben eine eigene Nummerierung für Bilder, Tabellen und Formeln.

n-GR-Gruppe 2

Zu dieser Gruppe gehören alle Polynome, deren Länge $n=2^q+r$, mit $r>0$ ist. **Abbildung 1** stellt die n-GR eines typischen Vertreters der Gruppe 2 dar, die 19-GR.

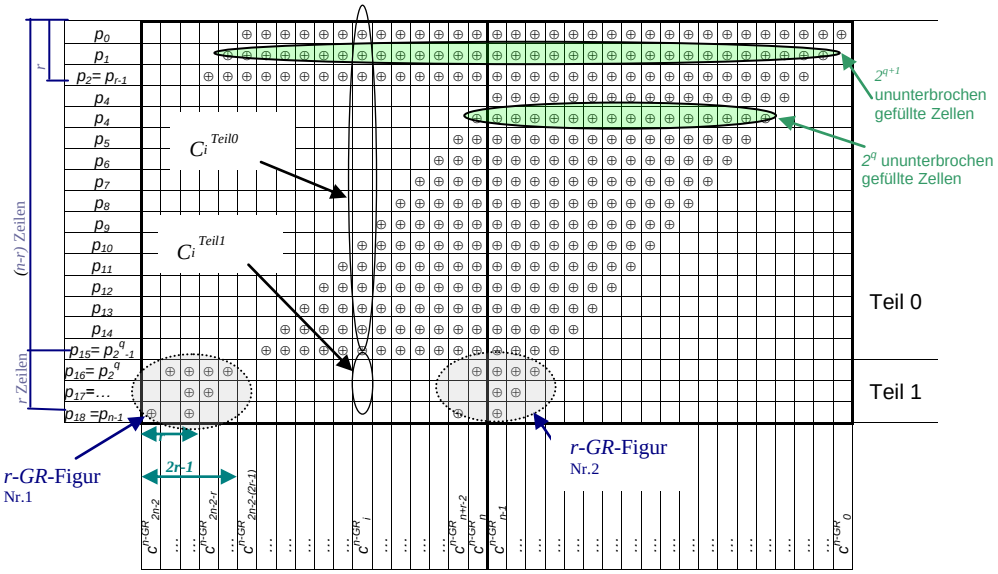


Abbildung 1: Vertreter der n-GR-Gruppe 2, 19-GR, $n=19=2^4+3 \Rightarrow q=4, r=3$

Die n-GR dieser Gruppe bestehen aus 2 Teilen:

- **Teil 0** besteht aus Zeilen $p_0 \ldots p_2{}^q{}_{-1}$, wobei jede Zeile $p_0 \ldots p_{r-1}$ $L=2^{q+1}$ ununterbrochen gefüllte Zellen hat, und jede Zeile $p_r \ldots p_2{}^q{}_{-1}$ $L=2^q$. Die Teil-Spalten, welche zum Teil 0 gehören, werden mit c_i^{Teil0} bezeichnet (siehe **Abbildung 1**).
- **Teil 1** besteht aus $p_2{}^q \ldots p_{n-1}$- Zeilen, die die zwei *r-GR*-Figuren, *r-GR*-Figur Nr.1 und die *r-GR*-Figur Nr.2 beinhalten (siehe **Abbildung 1**). Die *r-GR*-Figur Nr.1 befindet sich in den Spalten $c^{n\text{-}GR}{}_{2n-2}, \ldots, c^{n\text{-}GR}{}_{2n-2-(2r-1)}$. Die *r-GR*-Figur Nr.2 befindet sich in den Spalten $c^{n\text{-}GR}{}_{n-1-r+1}, \ldots, c^{n\text{-}GR}{}_{n-1+(r-1)}$, wobei sich ihre mittlere Spalte in der mittleren Spalte der ganzen *n*-GR befindet.

Zuerst wird der XOR- Aufwand des *voll iterativen* Ablaufplanes ermittelt, der in **Abbildung 2** dargestellt ist. In diesem Fall ist die linke n-GR-Spalte $c^{n\text{-}GR}{}_{2n-2}$, die aus nur einer gefüllten Zelle besteht, die Start-Spalte der Berechnung. Jede weitere rechts folgende Spalte $c^{n\text{-}GR}{}_{i-1}$ wird als die Summe (XOR) des Wertes der Spalte $c^{n\text{-}GR}{}_i$ und der Spalten-Differenz $\Delta\left(c_i^{n-GR}, c_{i-1}^{n-GR}\right)$ ermittelt.

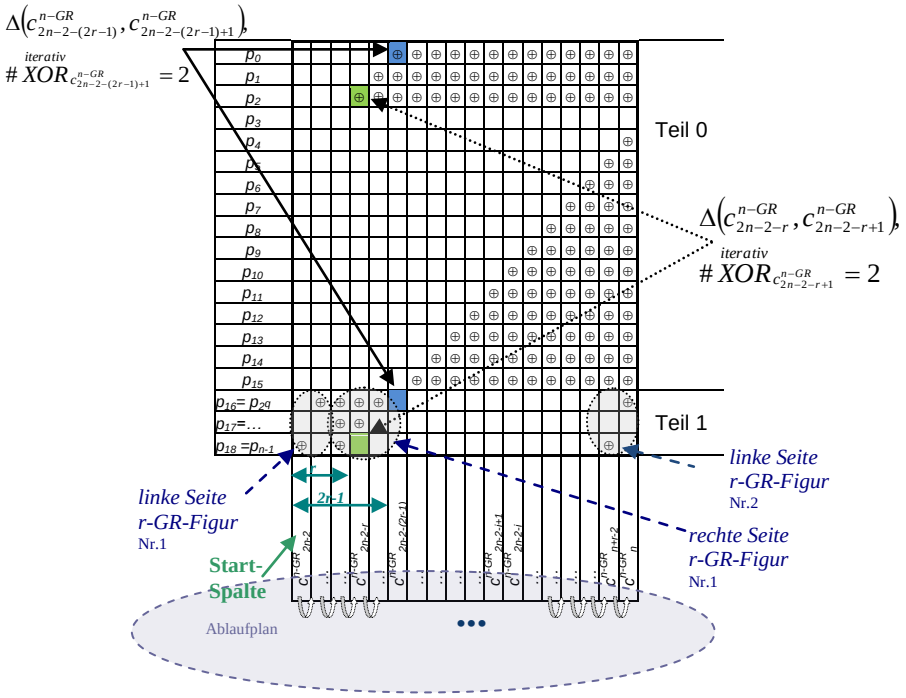


Abbildung 2: *voll iterativer* Ablaufplan der Berechnung der linken *19*-GR-Seite.

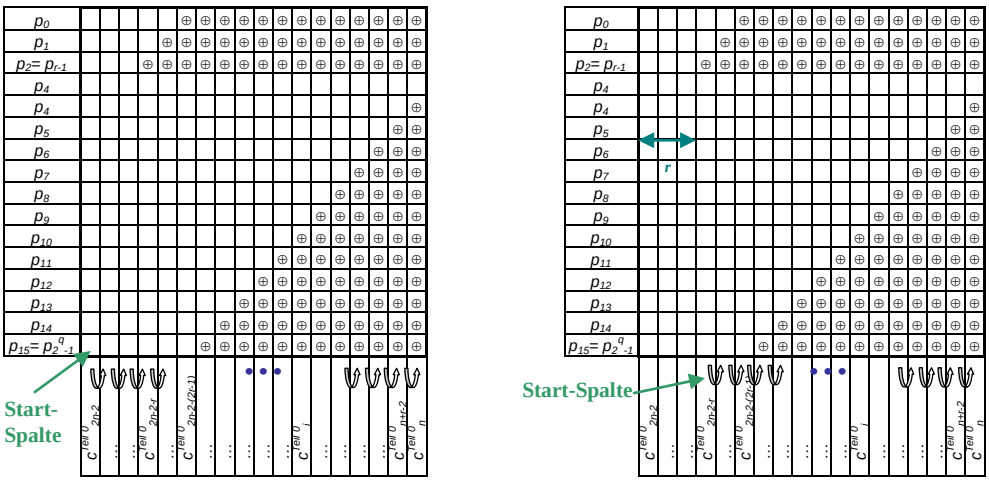
Diese TD besteht aus 2 Teilen: aus dem Teil 0 und aus dem Teil 1. Die letzten r Zeilen der TD bilden den Teil 1, in dem sich zwei r-GR-Figuren befinden. Der voll iterative Ablaufplan startet mit der separaten Berechnung der Spalte $c^{n\text{-}GR}_{2n\text{-}2}$. Die rechte Nachbar-Spalte wird als Summe des bereits ermittelten Wertes $c^{n\text{-}GR}_{2n\text{-}2}$ und der Spalten-Differenz $\Delta(c^{n\text{-}GR}_{2n\text{-}2}, c^{n\text{-}GR}_{2n\text{-}2\text{-}1})$ berechnet, usw. Grün gefärbte Zellen zeigen beispielweise, welche Teil-Produkte die Differenz-Menge der Spalten $c^{n\text{-}GR}_{2n\text{-}2\text{-}r+1}$ und $c^{n\text{-}GR}_{2n\text{-}2\text{-}r}$ bilden. Blau gefärbten Zellen zeigen, welche Teil-Produkte die Differenz-Menge der Spalten $c^{n\text{-}GR}_{2n\text{-}2\text{-}(2r\text{-}1)}$ und $c^{n\text{-}GR}_{2n\text{-}2\text{-}(2r\text{-}1)+1}$ bilden.

Formel **(1)** zeigt den XOR-Aufwand der *voll iterativen* Berechnung.

$$\#\,XOR^{voll\ iterativ}_{c^{n-GR}_{2n-2},\ldots,\mathcal{L}^{n-GR}_{n}} = \underbrace{\#\,XOR^{separat}_{c^{n-GR}_{2n-2}}}_{=\#\,XOR^{separat}_{c^{Teil\ 1}_{2n-2}}=0} + \#\,XOR^{voll\ iterativ}_{c^{Teil\ 1}_{2n-2},\ldots,\mathcal{L}^{Teil\ 1}_{n}} + \#\,XOR^{iterativ}_{c^{Teil\ 0}_{2n-2},\ldots,\mathcal{L}^{Teil\ 0}_{n}}$$

(1)

Entsprechend Formel **(1)** kann der XOR-Aufwand der *voll iterativen* Berechnung der linken *(n-1)* Spalten der *n*-GR als die Summe des XOR-Aufwandes der *voll iterativen* Berechnung des letzten GR-Teils (d.h. hier des Teils 1) und des XOR-Aufwandes der *iterativen* Berechnung des Restes, d.h. des Teils 0, berechnet werden. Nun werden diese Teile einzeln berechnet. **Abbildung 3-a)** zeigt die *iterative* und **Abbildung 3-b)** zeigt die *voll iterative* Berechnung des Teils.



a) iterative Berechnung des Teils 0:
die Berechnung beginnt mit der Ermittlung des Wertes der Start-Spalte $c^{Teil\ 0}_{2n-2}$

b) voll iterative Berechnung des Teils 0:
die Berechnung beginnt mit der Ermittlung des Wertes der Start-Spalte $c^{Teil\ 0}_{2n-2-r}$

Abbildung 3: Teil 0 der linken 19-GR-Seite.

Für die linke Seite des Teil 0 gilt folgendes: jede Spalte $c^{Teil\ 0}_{i}$, $i<2n-2-r$, beinhaltet alle TP ihrer linken Nachbar-Spalte und ein weiteres Teil-Produkt und kann als Summe (XOR) ihrer linker Spalte mit dem TP ermittelt werden. Der XOR-Aufwand dieser Berechnung beträgt nur 1 XOR-Gatter. Damit ist die *iterative* (evtl. *voll iterative*) Berechnung hier optimal. Der XOR-Aufwand der *voll iterativen* Berechnung des Teils 0 wird weiterhin mit $\#\ XOR^{voll\ iterativ}_{Teil\,0}$ bezeichnet, und der XOR-Aufwand der *iterativen* Berechnung des Teils 0 wird mit $\#XOR^{iterativ}_{Teil\,0}$ bezeichnet. Die beiden iterativen Fälle unterscheiden sich nur in der Berechnung der Spalte $c^{Teil\ 0}_{2n-2-r}$. Bei *iterativer* Berechnung wird für diese Spalte die Spalten-Differenz mit ihrer linken Spalte, d.h. mit dem Wert $c^{n-GR}_{2n-2-r+1}$, ermittelt. Bei der *voll iterativen* Berechnung ist diese Spalte die Start- Spalte und wird *separat* berechnet. Dies hat die folgende Auswirkung auf den XOR- Aufwand der Berechnung des Teils 0:

- *iterative* Berechnung nach dem Ablaufplan **(2)** hat den XOR-Aufwand **(3)**:

$$\underbrace{c^{separat}_{2n-2}}_{Start\text{–}Spalte}\ \overset{0}{\rightarrow}\ldots\overset{0}{\rightarrow}c^{Teil\ 0}_{2n-2-r+1}\overset{1}{\rightarrow}c^{Teil\ 0}_{2n-2-r}\overset{1}{\rightarrow}c^{Teil\ 0}_{2n-2-r-1}\overset{1}{\rightarrow}\ldots\overset{1}{\rightarrow}c^{Teil\ 0}_{n}$$

(2)

$$\#^{iterativ}XOR_{Teil\,0} = \#^{voll\ iterativ}XOR_{c_{2n-2}^{Teil\,0},\ldots,c_n^{Teil\,0}} = \underbrace{\#^{separat}XOR_{c_{2n-2}^{Teil\,0}}}_{=0} + \underbrace{\left|P_{\Delta\left(c_{2n-2}^{Teil\,0},c_{2n-2-1}^{Teil\,0}\right)}\right|}_{=0} + \ldots + \underbrace{\left|P_{\Delta\left(c_{2n-2-r+1}^{Teil\,0},c_{2n-2-r}^{Teil\,0}\right)}\right|}_{=1} + \ldots + \underbrace{\left|P_{\Delta\left(c_{n+1}^{Teil\,0},c_n^{Teil\,0}\right)}\right|}_{=1} = \sum_{i=r}^{n-2}1 = \underbrace{n}_{=2^q+r} - r - 1 = 2^q - 1$$

$$= \sum_{i=r}^{n-2}\underbrace{\left|P_{\Delta\left(c_{2n-2-i+1}^{Teil\,0},c_{2n-2-i}^{Teil\,0}\right)}\right|}_{=1}$$

(3)

- *voll iterative* Berechnung nach dem Ablaufplan **(4)** hat den XOR-Aufwand **(5)**:

$$\underbrace{c_{2n-2-r}^{separat\ Teil\,0}}_{Start\text{–}Spalte} \xrightarrow{1} c_{2n-2-r-1}^{Teil\,0} \xrightarrow{1} \ldots \xrightarrow{1} c_n^{Teil\,0}$$

(4)

$$\#^{voll\ iterativ}XOR_{Teil\,0} = \#^{voll\ iterativ}XOR_{c_{2n-2-r}^{Teil\,0},\ldots,c_n^{Teil\,0}} = \underbrace{\#^{separat}XOR_{c_{2n-2-r}^{Teil\,0}}}_{=0} +$$

$$+ \underbrace{\left|P_{\Delta\left(c_{2n-2-r}^{Teil\,0},c_{2n-2-r-1}^{Teil\,0}\right)}\right|}_{=1} + \ldots + \underbrace{\left|P_{\Delta\left(c_{n+1}^{Teil\,0},c_n^{Teil\,0l}\right)}\right|}_{=1} = \sum_{i=r+1}^{n-2}1 = \underbrace{n}_{=2^q+r} - r - 2 = 2^q - 2$$

$$= \sum_{i=r+1}^{n-2}\underbrace{\left|P_{\Delta\left(c_{2n-2-i+1}^{Teil\,0},c_{2n-2-i}^{Teil\,0}\right)}\right|}_{=1}$$

(5)

Jetzt wird der Teil 1 berechnet. **Abbildung 4** zeigt die *voll iterative* Berechnung des Teils 1.

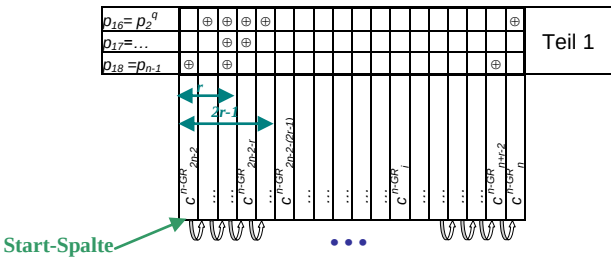


Abbildung 4: voll iterative Berechnung des Teils 1 der linken 19-GR-Seite.

Formel **(6)** zeigt den XOR-Aufwand dieser Berechnung.

$$\#\,XOR^{voll\ iterativ}_{Teil\,1} = \underbrace{\#\,XOR^{voll\ iterativ}_{\substack{linke\\ Seite\\ r-GR-Figur}} + \underbrace{\left(\#\,XOR^{iterativ}_{c_{r-1}^{r-GR}} + r\right)}_{\#XOR^{\substack{iterativ\\ rechte\\ Seite\\ r-GR-Figur}}}}_{r\text{-}GR\text{-Figur Nr.1}} + \underbrace{0+(r-1)}_{r\text{-}GR\text{-Figur Nr.2}} =$$

$$=\#\,XOR^{voll\ iterativ}_{\substack{linke\\ Seite\\ r-GR-Figur}} + \#\,XOR^{iterativ}_{c_{r-1}^{r-GR}} + 2r - 1 \qquad \textbf{(6)}$$

Formel **(6)** ist gültig für alle n der Gruppe 2 mit r>1. Im Fall *r=1* ist die Spalte $c^{1\text{-}GR}{}_0$ der linken *r-GR*-Figur die Start-Spalte der Berechnung und wird nicht *iterativ* gesucht. Aus den Formeln **(1)**, **(2)** und **(4)** ergibt sich unter Berücksichtigung des Sonderfalls *r=1* der folgende XOR-Aufwand der *voll iterativen* Berechnung der linken Seite der *n*-GR (d.h. der linken *n-1* Spalten):

$$\#\,XOR^{voll\ iterativ}_{c_{2n-2}^{n-GR},\ldots c_n^{n-GR}} = \#\,XOR^{voll\ iterativ}_{\text{Teil}\,1} + \#\,XOR^{iterativ}_{\text{Teil}\,0} =$$

$$= 2^q + \begin{cases} 0, & r=1 \\ \#\,XOR^{voll\ iterativ}_{\substack{linke\\ Seite\\ r-GR-Figur}} + XOR^{voll\ iterativ}_{c_{r-1}^{r-GR}} + 2r-2, & r>1 \end{cases} \qquad \textbf{(7)}$$

Der XOR-Aufwand der Berechnung der Spalte $c^{n\text{-}GR}{}_{n-1}$ aus dem Spalten-Wert $c^{n\text{-}GR}{}_n$ kann nach dem *voll iterativen* Ablaufplan folgendermaßen ermittelt werden:

$$\#XOR^{voll\ iterativ}_{c_{n-1}^{n-GR}} = \underbrace{\#XOR^{iterativ}_{c_{n-1}^{Teil\,0}}}_{=\left|P_\Delta\left(c_n^{Teil\,0}, c_{n-1}^{Teil\,0}\right)\right|=1} + \underbrace{\#XOR^{voll\ iterativ}_{c_{n-1}^{Teil\,1}}}_{\substack{=\#XOR_{\substack{\text{Addition bereits berechneter}\\ \text{Spalten-Differenz } \Delta(c_r^{r-GR}, c_{r-1}^{r-GR})\\ \text{mit dem Spalten-Wert } c_{n-1}^{Teil\,0}}} =1}} = 2 \qquad \textbf{(8)}$$

Die *voll iterative* Berechnung der linken *n*-GR-Seite wurde als Ablaufplan **A1** (siehe Kapitel 4.3.4.2) bezeichnet.

Tabelle 1 zeigt, welche Kombinationen der *iterativen* und *separaten* Berechnung der beiden GR-Teile weiter untersucht werden.

Tabelle 1: Untersuchte Ablaufpläne für n-GR-Gruppe 2

Teil 0	Teil 1			Bezeichnung des Ablaufplanes
	linke *r-GR*-Figur		rechte *r-GR*-Figur	
	linke Seite	rechte Seite	linke Seite	
iterativ	*voll iterativ*	*iterativ*	*iterativ*	**A1**
voll iterativ	*separat*	*separat*	*separat*	**A2**
iterativ	*separat*	*iterativ*	*separat*	**A3**
voll iterativ	*voll iterativ*	*separat*	*iterativ*	**A4**

In Kapitel 4.3.4.2 wurde bereits erklärt, dass nur die Ablaufpläne **A2** und **A3** verglichen werden müssen, wobei die Zuordnung der linken und rechten Seiten der

r-GR-Figur Nr.1 den optimalen Ablaufplan der Berechnung der linken *n*-GR-Seite bestimmt. Damit kann diese Zuordnung als Sonderfall betrachtet werden.
Jetzt wird der XOR- Aufwand des Ablaufplanes **A2** berechnet. **Abbildung 5** stellt den Ablaufplan dar.

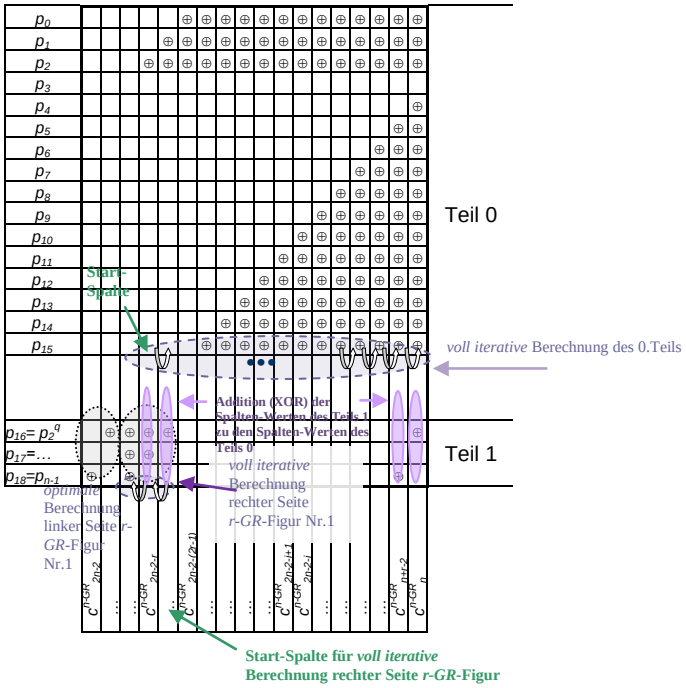


Abbildung 5: Ablaufplan **A2** der Berechnung der linken Seite der 19-GR.

Bei der *separaten* Berechnung der linken Seite der r-GR-Figur Nr.1 wird diese Seite der Figur als einzelne TD betrachtet, optimal berechnet und die berechneten Werte werden zu den jeweiligen GR-Spalten geXORt. Nun sind die linken *r* Spalten der linken *r-GR*-Figur der einzige Inhalt der jeweiligen GR-Spalten. Deswegen werden XOR-Gatter nur für die optimale Berechnung der linken Seite der Figur benötigt. Dieser XOR-Aufwand wird weiterhin mit $\#XOR^{optimal}_{\substack{linke\ Seite \\ r-GR-Figur}}$ bezeichnet. Die optimale Berechnung der rechten Seite (siehe Kapitel 4.3.4.1) der *r-GR*-Figur ist die *voll iterative* Berechnung, und ihr XOR-Aufwand beträgt *(r-1)* XOR-Gatter. Die rechten *(r-1)* Spalten der linken *r-GR*-Figur müssen aber geXORt werden. Dafür werden *(r-1)* XOR-Gatter benötigt. Der gesamte XOR-Aufwand der *separaten* Berechnung für den Fall ist wie folgt:

$$\#XOR^{A2}_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}} = \underbrace{\#XOR_{Teil\,0}^{voll\ iterativ}}_{=2^q-2} +$$

$$+ \underbrace{\underbrace{\#XOR^{separat\ linke}_{Seite\ r-GR-Figur}}_{=\#XOR^{optimal\ linke}_{Seite\ r-GR-Figur}} + \underbrace{\#XOR}_{=0}}_{r-GR-Figur\ \mathrm{Nr.1}} + \underbrace{\underbrace{\#XOR^{separat\ rechte}_{Seite\ r-GR-Figur}}_{=\#XOR^{optimal\ rechte}_{Seite\ r-GR-Figur}} + \underbrace{\#XOR}_{=r-1}}_{r-GR-Figur\ \mathrm{Nr.1}} + \underbrace{\underbrace{\#XOR^{separat\ linke}_{Seite\ r-GR-Figur}}_{=\#XOR^{optimal\ linke}_{Seite\ r-GR-Figur}} + \underbrace{\#XOR}_{=0}}_{r-GR-Figur\ \mathrm{Nr.2}} =$$

$$= n + 2r - 5 + \#XOR^{optimal\ linke}_{Seite\ r-GR-Figur}$$

(9)

Wenn die rechten *r* Spalten *iterativ* berechnet werden sollen, wie im Plan **A3** beschrieben, ist die n-GR-Spalte $c^{n\text{-}GR}{}_{2n\text{-}2\text{-}r+2}$, deren Inhalt dem Wert $c^{r\text{-}GR}{}_{r}$ gleich ist, die Start-Spalte. Der Wert der n-GR-Spalte $c^{GR}{}_{2n\text{-}2\text{-}r+1}$, deren Inhalt dem Spalten-Wert $c^{r\text{-}GR}{}_{r\text{-}1}$ der *r-GR*-Figur gleich ist, wird mittels Addition (XOR) der Spalten-Differenz $\Delta\left(c_r^{r-GR}, c_{r-1}^{r-GR}\right)$ zu dem Wert $c^{r\text{-}GR}{}_{r}$ ermittelt. Dieses erfolgt entsprechend dem optimalen Ablaufplan der Berechnung der linken Seite *r-GR*-Figur mit dem XOR-Aufwand $\#XOR^{optimal}_{c_{r-1}^{r-GR}}$. Formel **(10)** stellt den XOR-Aufwand der Berechnung nach dem Plan **A3** dar. **Abbildung 6** zeigt den Plan **A3** graphisch.

$$\#XOR^{A3}_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}} = \underbrace{\#XOR_{Teil\,0}^{iterativ}}_{=1+\#XOR_{Teil\,0}^{voll\ iterativ}} +$$

$$+ \underbrace{\underbrace{\#XOR^{separat\ linke}_{Seite\ r-GR-Figur}}_{=\#XOR^{optimal\ linke}_{Seite\ r-GR-Figur}} + \underbrace{\#XOR^{iterativ\ rechte}_{Seite\ r-GR-Figur}}_{=\#XOR^{optimal}_{c_{r-1}^{r-GR}} + \underbrace{\left|P_{\Delta\left(c_{r-1}^{r-GR}, c_{r-2}^{r-GR}\right)}\right|}_{=1} + \ldots + \underbrace{\left|P_{\Delta\left(c_1^{r-GR}, c_0^{r-GR}\right)}\right|}_{=1}}_{=r-1} + \underbrace{\left|P_{\Delta\left(c_{2n-2-(2r-1)+1}^{Teil\,1}, c_{2n-2-(2r-1)}^{Teil\,1}\right)}\right|}_{=1}}_{r-GR-Figur\ \mathrm{Nr.1}} + \underbrace{\underbrace{\#XOR^{separat\ linke}_{Seite\ r-GR-Figur}}_{=r-1}}_{r-GR-Figur\ \mathrm{Nr.2}} =$$

$$= n + r - 2 + \#XOR^{optimal\ linke}_{Seite\ r-GR-Figur} + \#XOR^{optimal}_{c_{r-1}^{r-GR}}$$

(10)

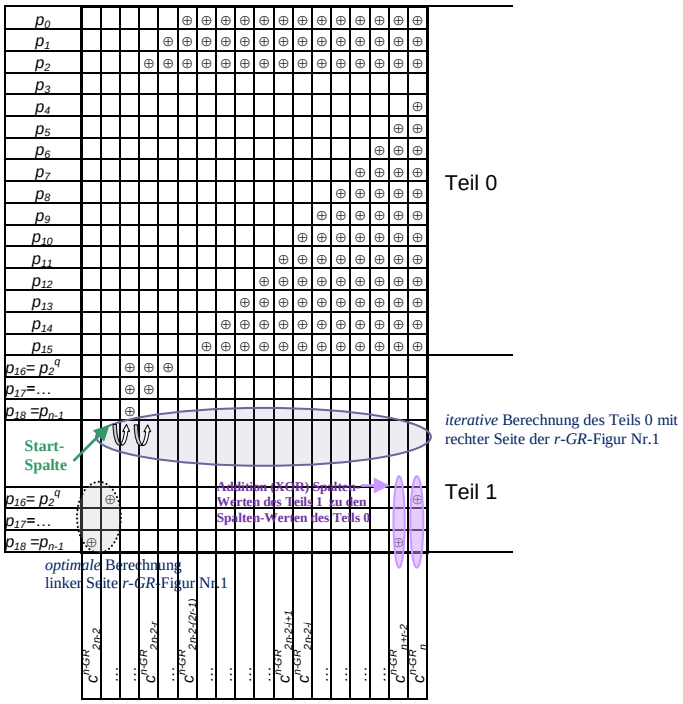


Abbildung 6: Ablaufplan **A3** der Berechnung der linken Seite der 19-GR.

Aus dem Vergleich von **(9)** und **(10)** ergibt sich folgendes: Wenn $r < \left(3 + \#^{optimal} XOR_{c_{r-1}^{r-GR}} \right)$, ist der Ablaufplan **A2** optimal, sonst Ablaufplan **A3**. Diese Ungleichung ist der Ungleichung $r \leq 3$ äquivalent[1]. Daraus folgt[2]:

- wenn $r \leq 3$, dann ist die *separate* Berechnung der linken *r-GR*-Figur *optimal*
- wenn $r \geq 4$, dann ist die *separate* Berechnung der linken Seite und die *iterative* Berechnung der rechten Seite der *r-GR*-Figur *optimal*

Der XOR-Aufwand der Berechnung des Spalten-Wertes $c^{n\text{-}GR}_{n\text{-}1}$ nach dem optimalen Ablaufplan ist für die beiden Sonderfälle gleich:

$$\#^{optimal} XOR_{c_{n-1}^{n-GR}} = \begin{cases} \#^{A2} XOR_{c_{n-1}^{n-GR}}, & r \leq 3 \\ \#^{A3} XOR_{c_{n-1}^{n-GR}}, & r \geq 4 \end{cases} = \underbrace{\left| P_{\Delta}\left(c_n^{Teil\,0}, c_{n-1}^{Teil\,0}\right) \right|}_{=1} + \underbrace{1}_{\#XOR\ \substack{\text{Addition der berechneten} \\ c_{r-1}^{r-GR} \\ mit\ c_{n-1}^{Teil\,0}}} = 2 \qquad \textbf{(11)}$$

[1] Es wurde festgestellt, dass für jede n-GR-Gruppe die folgende Ungleichung wahr ist: $\#^{optimal} XOR_{c_{r-1}^{r-GR}} \leq 3$.

[2] Unter Bedingungen, dass der Inhalt der Teile *0…(j-1) iterativ* (bzw. *voll iterativ*) berechnet werden muss und dass die *iterative* und *separate* Berechnungen des Restes des Teils *j* gleich aufwändig sind, kann diese Folgerung auf ähnliche Sonderfälle erweitert werden.

Tabelle 2 fasst die Ergebnisse kurz zusammen.

Tabelle 2: Optimale Ablaufpläne und deren XOR-Aufwände für die Berechnung linken Seite n-GR aus Gruppe 2

	Fall	Ablaufplan in **Tabelle 1**	XOR-Aufwand der Berechnung linker n-GR-Seite Ablaufplan $\#\ XOR\ c_{2n-2}^{n-GR}...c_{n}^{n-GR}$	XOR-Aufwand der Berechnung der Spalte $c^{n\text{-}GR}_{n-1}$ Ablaufplan $\#XOR_{c_{n-1}^{n-GR}}$
optimal	$r \le 3$	**A2**	$n+2r-5+\#\ XOR^{optimal\ linke\ Seite}_{r-GR-Figur} = n+2r-5$	2
	$r \ge 4$	**A3**	$n+r-2+\#\ XOR^{optimal\ linke\ Seite}_{r-GR-Figur} + \#\ XOR^{optimal}_{c_{r-1}^{r-GR}}$	2
voll iterativ	$r=1$	**A1**	$2^q = n-r$	2
	$r>1$	**A1**	$\#\ XOR^{voll\ iterativ\ linke\ Seite}_{r-GR-Figur} + \#\ XOR^{voll\ iterativ}_{c_{r-1}^{r-GR}} + 2^q + 2r - 2 =$ $= \#\ XOR^{voll\ iterativ\ linke\ Seite}_{r-GR-Figur} + \#\ XOR^{voll\ iterativ}_{c_{r-1}^{r-GR}} + n + r - 2$	2

Nun werden die Ablaufpläne und deren XOR- Aufwande der *nicht vollen* Großen Rauten *NV-n*-GR der n-GR-Gruppe 2 untersucht. Der Unterschied zu den *n*-GR ist nur, dass die *n-GR*-Zeile mit dem TP p_{n-1} bei der *NV-n*-GR leer ist. Aus diesem Grund besteht die linke Seite der *NV-r-GR*-Figur nicht aus (*r-1*), sondern aus (*r-2*) Spalten, und an der rechten Seite sind die Spalten $c^{NV\text{-}r\text{-}GR}{}_{r}$ und $c^{NV\text{-}r\text{-}GR}{}_{r-1}$ gleich. Der Teil 1 beinhaltet jetzt zwei *NV-r-GR*-Figuren. **Abbildung 7** zeigt das.

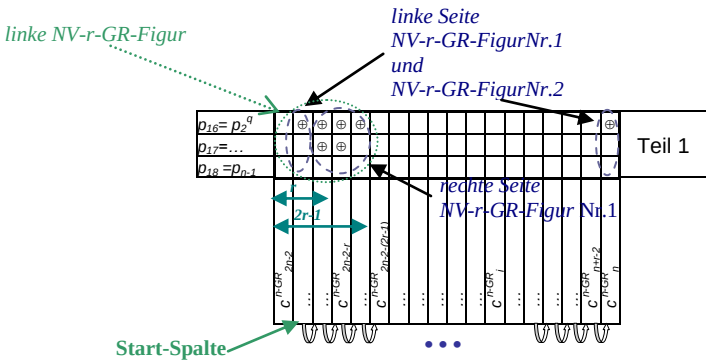


Abbildung 7: *Voll iterative* Berechnung des Teils 1 der linken Seite der NV-19-GR.

Aus den gleichen Gründen, wie bei *n-GR*, werden die in **Tabelle 1** aufgelisteten Ablaufpläne untersucht. Bei dem Sonderfall *r=1* fehlt der ganze Teil 1 der *NV-n-GR* und der XOR-Aufwand ergibt sich nur aus der *voll iterativen* Berechnung des Teils 0 (siehe **(5)**).
Formel **(12)** zeigt den XOR-Aufwand der *voll iterativen* Berechnung der *NV-n-GR* für *r>1*.

$$\begin{aligned}
&\#^{voll\ iterativ}XOR_{c_{2n-3}^{NV-n-GR},\ldots,c_n^{NV-n-GR}} = \#^{A1}XOR_{c_{2n-3}^{NV-n-GR},\ldots,c_n^{NV-n-GR}} = \#^{voll\ iterativ}XOR_{NV-n-GR}^{Teil\,1} + \#^{iterativ}XOR_{Teil\,0} = \\
&= \#^{voll\ iterativ}XOR_{NV-r-GR-Figur}^{linke\ Seite} + \underbrace{\#^{iterativ}XOR_{NV-r-GR-Figur}^{rechte\ Seite}}_{=\#^{voll\ iterativ}XOR_{c_{r-1}^{NV-r-GR}+r-2}} + \underbrace{\left|P_{\Delta}\left(c_{2n-2-(2r-1)+1}^{Teil\,1\ NV-n-GR}, c_{2n-2-(2r-1)}^{Teil\,1\ NV-n-GR}\right)\right|}_{=1} + \underbrace{0+(r-2)}_{NV-r-GR-\text{FigurNr.2}} + \\
&\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{NV-r-GR-\text{FigurNr.1}} \\
&\underbrace{\qquad}_{\#^{voll\ iterativ}XOR_{NV-n-GR}^{Teil\,1}} \\
&+ \underbrace{2^q-1}_{\#^{iterativ}XOR_{0.Teils}} = \#^{voll\ iterativ}XOR_{NV-r-GR-Figur}^{linke\ Seite} + \#^{voll\ iterativ}XOR_{c_{r-1}^{NV-r-GR}} + 2^q + 2r - 4 = \\
&= \#^{voll\ iterativ}XOR_{NV-r-GR-Figur}^{linke\ Seite} + \#^{voll\ iterativ}XOR_{c_{r-1}^{NV-r-GR}} + n + r - 4
\end{aligned}$$

(12)

Der XOR-Aufwand der Berechnung des Spalten-Wertes $c^{NV\text{-}n\text{-}GR}{}_n$ nach dem *voll iterativen* Ablaufplan kann folgendermaßen ermittelt werden:

$$\#^{voll\ iterativ}XOR_{c_{n-1}^{NV-n-GR}} = \underbrace{\#^{A1}XOR_{c_{n-1}^{Teil\,0}}}_{=\left|P_{\Delta}\left(c_n^{Teil\,01}, c_{n-1}^{Teil\,0}\right)\right|} + \begin{cases} 0, r=1 \\ \underbrace{\#^{voll\ iterativ}XOR_{c_{n-1}^{Teil\,1}}, r>1}_{=\#XOR_{\substack{\text{Addition bereits berechneter} \\ \text{Spalten-Differenz } \Delta(c_r^{NV-r-GR}, c_{r-1}^{NV-r-GR}) \\ \text{mit dem Spalten-Wert } c_{n-1}^{Teil\,0}}}} \end{cases} = 1 + \begin{cases} 0, r=1 \\ 1, r>1 \end{cases}$$

(13)

Ähnlich wie bei der vollen *n-GR* können die Pläne **A2** und **A3** für *NV-n-GR* untersucht werden. Bei der *separaten* Berechnung der linken Seite der *NV-r-GR*-Figur (*r>1*) wird diese Seite als einzelne TD betrachtet, optimal berechnet und die berechneten Werte werden zu den entsprechenden GR-Spalten geXORt. Nun sind die linken *r-1* Spalten der linken *NV-r-GR*-Figur der einzige Inhalt der jeweiligen GR-Spalten. Deswegen werden die XOR-Gatter nur für die optimale Berechnung der linken Seite der Figur benötigt. Dieser XOR-Aufwand wird weiterhin mit $\#^{optimal}XOR_{NV-r-GR-Figur}^{linke\ Seite}$ bezeichnet. Die optimale Berechnung der rechten Seite (siehe Kapitel 4.3.4.1) der *r-GR*-Figur ist die *voll iterative* Berechnung, und ihr XOR-Aufwand beträgt *(r-2)* XOR-Gatter. Die rechten *(r-2)* Spalten der linken *r-GR*-Figur müssen aber geXORt werden. Dafür werden *(r-2)* XOR-Gatter benötigt. Der gesamte XOR-Aufwand der *separaten* Berechnung im Fall *r>1* ist wie folgt:

$$\#XOR^{A2}_{c_{2n-3}^{NV-n-GR}\ldots c_n^{NV-n-GR}} = \underbrace{\#\,XOR^{voll\ iterativ}_{Teil\,0}}_{=2^q-2} +$$

$$+ \underbrace{\#XOR^{separat\ linke\ Seite}_{NV-r-GR-Figur}}_{\underbrace{=\#XOR^{optimal\ linke\ Seite}_{NV-r-GR-Figur}}_{}\ +\underbrace{\#XOR^{Addition\ der\ berechneten\ Spalten-Werten\ mit\ den\ entsprechenden\ Spalten-Werten\ des\ Teils\ 0}}_{=0}} + \underbrace{\#XOR^{separat\ rechte\ Seite}_{NV-r-GR-Figur}}_{\underbrace{=\#XOR^{optimal\ rechte\ Seite}_{NV-r-GR-Figur}}_{=r-2}\ +\underbrace{\#XOR^{Addition\ der\ berechneten\ Spalten-Werten\ mit\ den\ entsprechenden\ Spalten-Werten\ des\ Teils\ 0}}_{=r-1}} + \underbrace{\#XOR^{separat\ linke\ Seite}_{NV-r-GR-Figur}}_{\underbrace{=\#XOR^{optimal\ linke\ Seite}_{NV-r-GR-Figur}}_{=0}\ +\underbrace{\#XOR^{Addition\ der\ berechneten\ Spalten-Werten\ mit\ den\ entsprechenden\ Spalten-Werten\ des\ Teils\ 0}}_{=r-2}} =$$

$$\underbrace{\underbrace{\underbrace{\#XOR^{separat\ linke\ Seite}_{NV-r-GR-Figur}}_{=\#XOR^{optimal\ linke\ Seite}_{NV-r-GR-Figur}}\ +\underbrace{\#XOR^{Addition\ der\ berechneten\ Spalten-Werten\ mit\ den\ entsprechenden\ Spalten-Werten\ des\ Teils\ 0}}_{=0}}_{NV-r-GR-Figur\,\mathrm{Nr.1}} + \underbrace{\underbrace{\#XOR^{separat\ rechte\ Seite}_{NV-r-GR-Figur}}_{=\#XOR^{optimal\ rechte\ Seite}_{NV-r-GR-Figur}}\ +\underbrace{\#XOR^{Addition\ der\ berechneten\ Spalten-Werten\ mit\ den\ entsprechenden\ Spalten-Werten\ des\ Teils\ 0}}_{=r-1}}_{NV-r-GR-Figur\,\mathrm{Nr.1}}}_{\#XOR^{separat\ Teil\ 1}_{NV-n-GR}} + \underbrace{\underbrace{\underbrace{\#XOR^{separat\ linke\ Seite}_{NV-r-GR-Figur}}_{=\#XOR^{optimal\ linke\ Seite}_{NV-r-GR-Figur}}\ +\underbrace{\#XOR^{Addition\ der\ berechneten\ Spalten-Werten\ mit\ den\ entsprechenden\ Spalten-Werten\ des\ Teils\ 0}}_{=r-2}}_{NV-r-GR-Figur\,\mathrm{Nr.2}}}_{\#XOR^{separat\ Teil\ 1}_{NV-n-GR}}$$

$$= n + 2r - 7 + \#XOR^{optimal\ linke\ Seite}_{NV-r-GR-Figur}$$

(14)

Formel **(15)** zeigt den XOR-Aufwand der Berechnung nach dem Plan **A3** im Fall *r>1*.

$$\#XOR^{A3}_{c_{2n-2}^{NV-n-GR}\ldots c_n^{NV-n-GR}} =$$

$$= \underbrace{\#\,XOR^{iterativ}_{Teil\,0}}_{=1+\#\,XOR^{voll\ iterativ}_{Teil\,0}} + \underbrace{\left(\begin{array}{l} \underbrace{\underbrace{\#XOR^{separat\ linke\ Seite}_{NV-r-GR-Figur}}_{=\#XOR^{optimal\ linke\ Seite}_{NV-r-GR-Figur}} + \underbrace{\#XOR^{iterativ\ rechte\ Seite}_{NV-r-GR-Figur}}_{\underbrace{=\#XOR^{optimal}_{c_{r-1}^{NV-r-GR}}}_{}+\underbrace{\left|P_{\Delta\left(c_{r-1}^{NV-r-GR},c_{r-2}^{NV-r-GR}\right)}\right|}_{=0}+}} + \\ \underbrace{\underbrace{+\left|P_{\Delta\left(c_{r-2}^{NV-r-GR},c_{r-3}^{NV-r-GR}\right)}\right|}_{=1}+\ldots+\underbrace{\left|P_{\Delta\left(c_1^{NV-r-GR},c_0^{NV-r-GR}\right)}\right|}_{=1}}_{=r-2} \\ \underbrace{+\underbrace{\left|P_{\Delta\left(c_{2n-2-(2r-1)+1}^{Teil\ 1\ n-GR},c_{2n-2-(2r-1)}^{1.Teil\ n-GR}\right)}\right|}_{=1} + \underbrace{\#XOR^{separat\ linke\ Seite}_{NV-r-GR-Figur}}_{=r-2}}_{NV-r-GR-Figur\,\mathrm{Nr.2}} \end{array}\right)}_{\#XOR^{A3}_{Teil\,1\ NV-n-GR}} =$$

$$= n + r - 4 + \#XOR^{optimal\ linke\ Seite}_{NV-r-GR-Figur} + \#XOR^{optimal}_{c_{r-1}^{NV-r-GR}}$$

(15)

Aus dem Vergleich von **(14)** und **(15)** ergibt sich folgendes:

Wenn $r < \left(3 + \#XOR^{optimal}_{c_{r-1}^{NV-r-GR}}\right)$, ist der Ablaufplan **A2** optimal, sonst der Ablaufplan **A3**.

Diese Ungleichung ist der Bedingung $r \le 3$ äquivalent.

Der XOR-Aufwand der Berechnung der Spalten $c^{NV\text{-}n\text{-}GR}_{n\text{-}1}$ nach dem optimalen Ablaufplan ist wie folgt:

$$\#XOR^{optimal}_{c^{NV-n-GR}_{n-1}} = \begin{cases} \#XOR^{A1}_{c^{NV-n-GR}_{n-1}}, r=1 \\ \#XOR^{A2}_{c^{NV-n-GR}_{n-1}}, 1<r\le 3 \\ \#XOR^{A3}_{c^{NV-n-GR}_{n-1}}, r\ge 4 \end{cases} = \begin{cases} \underbrace{\left|P_{\Delta\left(c_n^{Teil\,0}, c_{n-1}^{Teil\,0}\right)}\right|}_{=1}, & r=1 \\ \underbrace{\left|P_{\Delta\left(c_n^{Teil\,0}, c_{n-1}^{Teil\,0}\right)}\right|}_{=1} + \underbrace{1}_{\substack{Addition \\ der\ berechneten \\ \#XOR_{c_{r-1}^{NV-r-GR}} \\ mit\ c_{n-1}^{Teil\,0}}}, & r>1 \end{cases} = \begin{cases} 1, r=1 \\ 2, r>1 \end{cases}$$

(16)

Tabelle 3 fasst die Ergebnisse kurz zusammen.

Tabelle 3: Optimale Ablaufpläne und deren XOR-Aufwände für die Berechnung linken Seite NV-n-GR aus Gruppe 2

	Fall	Ablaufplan in **Tabelle 1**	XOR-Aufwand der Berechnung linker NV-n-GR-Seite $\#XOR^{Ablaufplan}_{c_{2n-3}^{NV-n-GR}..c_n^{NV-n-GR}}$	XOR-Aufwand der Berechnung der Spalte $c^{NV\text{-}n\text{-}GR}_{n\text{-}1}$ $\#XOR^{Ablaufplan}_{c_{n-1}^{NV-n-GR}}$
optimal	$r=1$	**A1**	$2^q-2=n-3$	1
	$r\in\{2,3\}$	**A2**	$n+2-7+\#XOR^{optimal}_{\substack{linke\ Seite \\ NV-r-GR-Figur}} = \begin{cases} n-3, & r=2 \\ n-1, & r=3 \end{cases}$	2
	$r\ge 4$	**A3**	$n+r-4+\#XOR^{optimal}_{\substack{linke\ Seite \\ NV-r-GR-Figur}} + \#XOR^{optimal}_{c_{r-1}^{NV-r-GR}}$	2
voll iterativ	$r=1$	**A1**	$2^q-2=n-3$	1
	$r>1$	**A1**	$n+r-4+\#XOR^{voll\ iterativ}_{\substack{linke\ Seite \\ NV-r-GR-Figur}} + XOR^{voll\ iterativ}_{c_{r-1}^{NV-r-GR}}$	2

n-GR-Gruppe 3

Zu dieser Gruppe gehören alle Polynome, deren Länge $n=2^q+2^{q-1}+\ldots+2^{q-j}$, mit $0<j\le q$ ist. **Abbildung 8** stellt die GR eines typischen Vertreters der Gruppe 3 dar, die 28-GR.

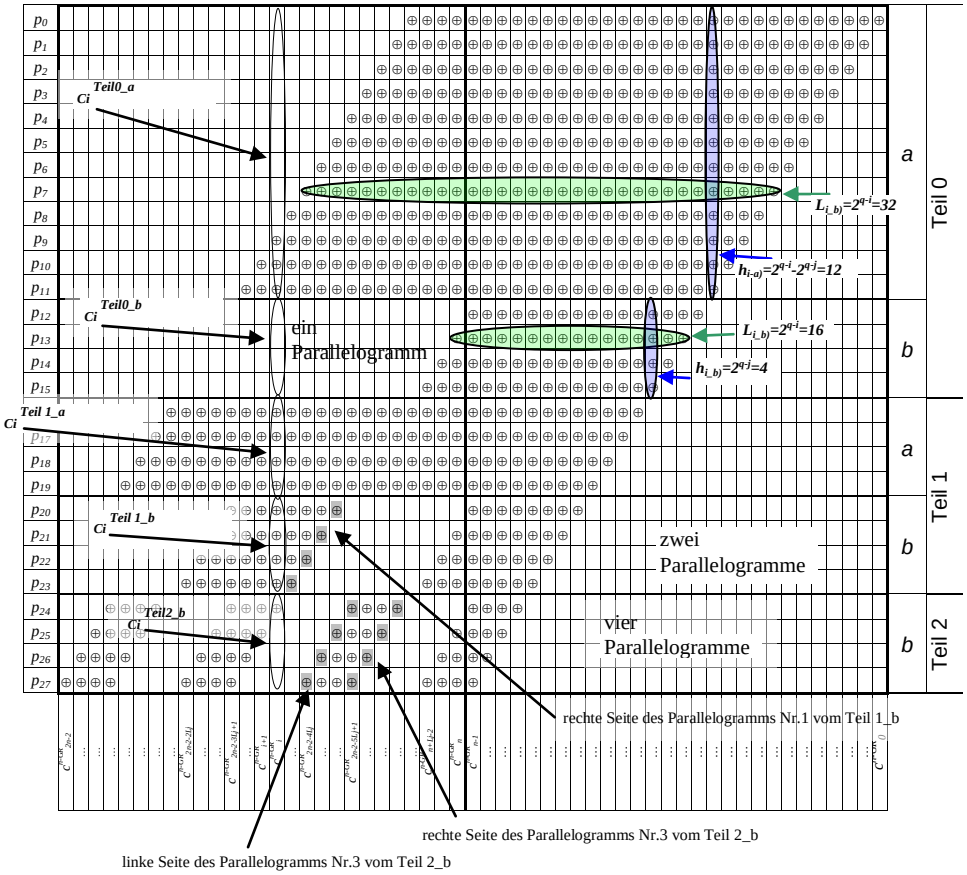


Abbildung 8: Vertreter der n-GR-Gruppe 3, *28-GR, n=28=11100$_2$ ⇒ q=4, j=2*

Die GR dieser n-GR-Gruppe bestehen aus (*j+1*) Teilen:

• Jeder **Teil *i*** (*0≤ i≤ j*) besteht aus 2^{q-i} Zeilen und kann in zwei weitere Sub-Teile folgendermaßen aufgeteilt werden:

- **Teil *i_a*** hat $(2^{q-i}-2^{q-j})$ Zeilen mit $L_{i\_a}=2^{q+1}$ ununterbrochen gefüllten Zellen, die ein Parallelogramm bilden
- **Teil *i_b*** besteht aus 2^{q-j} Zeilen, in denen sich 2^{i} gleiche Parallelogramme befinden. Diese Parallelogramme werden von links mit ‚1' beginnend durchnummeriert. Die Grundseite jedes Parallelogramms besteht aus $L_{i\_b}=2^{q-i}$ ununterbrochen gefüllten Zellen. Die dazugehörige Höhe entspricht $h_{i\_b}=2^{q-j}$ gefüllten Zellen. Die mittlere Spalte des rechtesten Parallelogramms ist ein Teil der mittleren Spalte der GR. Der Abstand zwischen allen Parallelogrammen ist gleich und besteht aus der folgenden Anzahl leerer Zellen: $\frac{2^{q+1}-2^{i}\cdot 2^{q-i}}{2^{i}}=2^{q-i}$

Bei dem **Teil *j*** besteht der **Teil *j_a*** aus $(2^{q-j}-2^{q-j})=0$ Zeilen, d.h. dass der **Teil *j*** nur den **Teil *j_b*** beinhaltet. Die Anzahl ununterbrochen gefüllter Zellen vom **Teil *j*** wird weiter mit L_j bezeichnet und die dazugehörige Höhe mit h_j.

Für jeden **Teil *i_a***, $0 \leq i<j$, gilt folgendes: jede Spalte dieses Teils beinhaltet in sich alle TP ihrer linken Nachbar-Spalte und kann ein weiteres Teil-Produkt beinhalten. Analog zur Gleichung **(121)** (siehe Kapitel 4.3.4.2) ist hier die *iterative* (evtl. *voll iterative*) Berechnung optimal. Auf alle anderen n-GR-Teile mit Ausnahme des Parallelogramms Nr.1 vom **Teil *j*** sind jetzt die Kriterien (***K***) anwendbar: alle GR-Teile mit $L>1$ gehören zu dem *iterativen* TD-Teil.
Für alle Parallelogramme jedes **Teils *i_b***, $1 \leq i<j$, gilt folgendes: Die linken Seiten aller Parallelogramme ab Nummer 2 befinden sich direkt unter den Seiten aller Parallelogramme des **Teils *(i-1)_b*** (siehe **Abbildung 8**). Diese Tatsache begünstigt den *iterativen* Ablaufplan der Berechnung, weil die Ermittlung der Spalten-Differenzen mithilfe bereits ermittelter Spalten-Differenzen besonders günstig durchgeführt werden kann.
Die Zuordnung des Parallelogramms Nr.1 vom **Teil *j*** zu dem *iterativen* oder dem *separaten* TD-Teil wird als Sonderfall betrachtet. Dies ist notwendig, da dieses Parallelogramm der einzige Inhalt der Spalten $c^{n\text{-}GR}{}_{2n-2}$, …, $c^{n\text{-}GR}{}_{2n-2-Lj+1}$ ist[3]. Diese Tatsache begünstigt die *separate* Berechnung des Parallelogramms Nr.1 und damit die separate Berechnung des ganzen Teils *j.*
Die Zuordnung des ganzen **Teils *j*** wird im Fall $L_j=1$ auch als Sonderfall betrachtet. Nach den Kriterien (***K***) (siehe Kapitel 3.9) soll er dem *separaten* TD-Teil zugeordnet werden. Unter Berücksichtigung der oben genannten Lage der Parallelogramm-Seiten muss jedoch auch die *iterative* Berechnung untersucht werden.
Tabelle 4 zeigt diese Sonderfälle und deren Ablaufpläne.

Tabelle 4: Untersuchte Ablaufpläne für n-GR-Gruppe 3

Fall	Teil *0…(j-1)_a*	Teil *(j-1)_b*	Teil *j*		Bezeichnung des Ablaufplanes
			Parallelogramm Nr.1	Parallelogramm Nr.2…Nr.2^j	
$L_j>2 \Leftrightarrow j \leq q-2$	*iterativ*	*iterativ*	*voll iterativ*	*iterativ*	**A1**
$L_j=2 \Leftrightarrow j= q-1$	*iterativ*	*iterativ*	*voll iterativ*	*iterativ*	**A1**
	voll iterativ	*iterativ*	*separat*	*iterativ*	**A2**
	voll iterativ	*iterativ*	*separat*	*separat*	**A3**
$L_j=1 \Leftrightarrow j= q$	*voll iterativ*	*iterativ*	*separat*	*separat*	**A3**
	voll iterativ	*separat* unter der Bedingung, dass die gefüllten Zellen von diesem Teil zusammen mit den gefüllten Zellen vom Teil j mehrere gleiche Figuren bilden	*separat*	*separat*	**A4**

Es wird mit der *voll iterativen* Berechnung der linken *n-GR*-Seite begonnen, was dem Plan **A1** (siehe **Tabelle 4**) entspricht. Der XOR-Aufwand dieses Ablaufplans kann folgendermaßen dargestellt werden:

[3] Die Kriterien ***(K)*** sind nicht anwendbar.

$$\#\ \overset{voll\ iterativ}{XOR}_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}} = \sum_{i=0}^{j-1} \#\ \overset{iterativ}{XOR}_{i\_a}^{Teil} + \sum_{i=0}^{j-1} \#\ \overset{iterativ}{XOR}_{i\_b}^{Teil} + \#\ \overset{voll\ iterativ}{XOR}_{Teil\ j}$$

(17)

Jetzt wird jede Komponente von **(17)** einzeln ausgeführt. Formel **(18)** zeigt den XOR-Aufwand der *iterativen* Berechnung aller Teile *i_a*.

$$\sum_{i=0}^{j-1} \#\ \overset{iterativ}{XOR}_{i\_a}^{Teil} = \sum_{i=0}^{j-1} \left(2^{q-i} - 2^{q-j}\right) = 2^{q+1} - 2^{q-j}(2+j)$$

(18)

Die *voll iterative* Berechnung dieser Teile, die für die Pläne **A2-A4** angewendet wird, benötigt wegen der *separaten* Berechnung der Start-Spalte 1 XOR-Gatter weniger:

$$\#\ \overset{voll\ iterativ}{XOR}_{\substack{0\_a \\ \ldots \\ (j-1)\_a}}^{Teil} = \sum_{i=0}^{j-1} \#\ \overset{iterativ}{XOR}_{i\_a}^{Teil} - 1 = 2^{q+1} - 2^{q-j}(2+j) - 1$$

(19)

Jetzt wird die *voll iterative* Berechnung aller Teile *i_b* ausgeführt. Die Lage der Parallelogramme begünstigt die *iterative* Berechnung dieser Teile, weil die Spalten-Differenzen selber *iterativ*, d.h. mittels bereits ermittelter Spalten-Differenzen, berechnet werden können. **Abbildung 9** illustriert das.

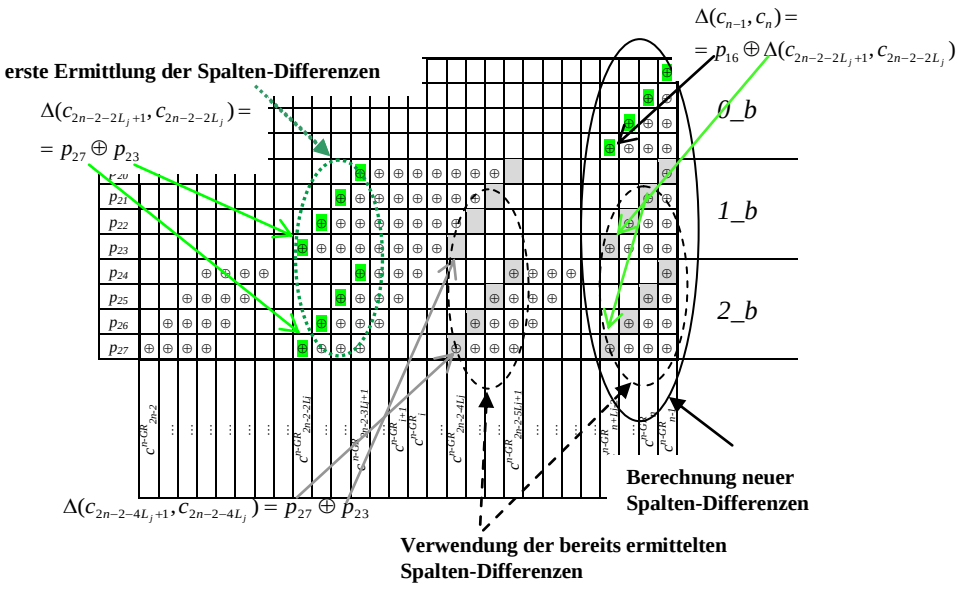


Abbildung 9: Alle Teile *i_b* der linken *n* Spalten der *28-GR*; iterative Berechnung der Spalten-Differenz

Bei der Berechnung des Spalten-Wertes $c^{n\text{-}GR}{}_{2n\text{-}2\text{-}2Lj}$ wird die Spalten-Differenz $\Delta(c^{GR}{}_{2n\text{-}2\text{-}2Lj+1}, c^{GR}{}_{2n\text{-}2\text{-}2Lj})$ ermittelt. In dem Beispiel-Fall der 28-GR ist sie wie folgt:

$$\Delta(c_{2n-2-2L_j+1}, c_{2n-2-2L_j}) = \Delta(c_{47}, c_{46}) = p_{27} \oplus p_{23}$$

Diese erste Ermittlung der Spalten-Differenz ist in **Abbildung 9** grün gekennzeichnet.

Bei der Ermittlung des Spalten-Wertes $c^{n\text{-}GR}{}_{2n\text{-}2\text{-}4Lj}$ wird die Spalten-Differenz $\Delta(c^{GR}{}_{2n\text{-}2\text{-}4Lj+1}, c^{GR}{}_{2n\text{-}2\text{-}4Lj})$ berechnet. Diese Berechnung kann gespart werden, da diese Spalten-Differenz dem bereits ermittelten Wert $\Delta(c^{GR}{}_{2n\text{-}2\text{-}2Lj+1}, c^{GR}{}_{2n\text{-}2\text{-}2Lj})$ gleich ist:

$$\Delta(c_{2n-2-4L_j+1}, c_{2n-2-4L_j}) = \Delta(c_{39}, c_{38}) = p_{27} \oplus p_{23} = \Delta(c_{47}, c_{46})$$

Derselbe – bereits ermittelte – Wert der Spalten-Differenz wird auch bei der Ermittlung der Spalten-Differenz $\Delta(c^{GR}{}_{2n\text{-}2\text{-}6Lj+1}, c^{GR}{}_{2n\text{-}2\text{-}6Lj})$ genutzt. In unserem Beispiel:

$$\Delta(c_{2n-2-4L_j+1}, c_{2n-2-4L_j}) = \Delta(c_{31}, c_{30}) = p_{14} \oplus p_{27} \oplus p_{23} = p_{14} \oplus \Delta(c_{47}, c_{46})$$

Die gesparten Ermittlungen der Spalten-Differenzen sind in **Abbildung 9** grau gekennzeichnet.

Die Möglichkeit, die Spalten-Differenzen *iterativ* mittels bereits ermittelter Spalten-Differenzen zu berechnen, kann nur für die linken Seiten der Parallelogramme angewendet werden. Diese Tatsache kann auch folgendermaßen umformuliert werden:

Die zwei linken Parallelogramme jedes Teils *i_b,* mit $2 \le i \le j$, tragen mit ihren beiden Seiten zum XOR-Aufwand bei, und jedes weitere Parallelogramm trägt nur mit seiner rechten Seite bei. Formel **(20)** zeigt den XOR-Aufwand der *voll iterativen* Berechnung aller Teile *i_b*.

$$\#\,XOR^{voll\ iterativ}_{\substack{Teil\ 0\_b \\ \dots \\ j\_b}} = \sum_{i=0}^{j-1} \#\,XOR^{iterativ}_{i\_b} + \underbrace{\#\,XOR^{voll\ iterativ}_{Teil\ j}}_{\#\,XOR^{iterativ}_{Teil\ j}-1} = \#\,XOR^{iterativ}_{0\_b} + \#\,XOR^{iterativ}_{1\_b} + \sum_{i=2}^{j} \#\,XOR^{iterativ}_{i\_b} - 1 =$$

$$= \underbrace{\left(2^{q-j}-1\right)}_{\substack{\text{Teil } 0\_b, \\ \text{nur 1 Parallelogramm,} \\ \text{linken } \left(2^{q-j}-1\right) \text{Spalten}}} + \underbrace{\left(\underbrace{2 \cdot 2^{q-j}}_{\text{Parallelogramm Nr.1}} + \underbrace{2^{q-j}-1}_{\substack{\text{linke} \left(2^{q-j}-1\right) \text{Spalten} \\ \text{des Parallelogramms Nr.2}}} \right)}_{\substack{\text{Teil } 1\_b, \\ \text{2 Parallelogramme}}} + \underbrace{\sum_{i=2}^{j} \left(\underbrace{\underbrace{2 \cdot 2^{q-j}}_{\text{Parallelogramm Nr.1}} + \underbrace{2 \cdot 2^{q-j}}_{\text{Parallelogramm Nr.2}} + \underbrace{\left(2^{i}-3\right) \cdot 2^{q-j}}_{\substack{\text{rechte Seiten anderer} \\ \text{Parallelogramme}}}}_{} \right)}_{\substack{\text{Teile } 2\_b \dots j\_b, \\ \text{mind. 4 Parallelogramme}}} - 1$$

(20)

Nach Formel **(20)** unterscheidet sich der XOR-Aufwand der Berechnung des Teils *0_b* und des Teils *1_b* von dem XOR-Aufwand der Berechnung anderer Teile *i_b* mit $j \ge 2$. Der Fall, in dem die *n*-GR nur aus dem Teil 0 und aus dem Teil 1 besteht, kann als Sonderfall mit $j=1$ betrachtet werden (siehe Formel **(21)**).

$$\#\ XOR^{voll\ iterativ}_{\substack{Teil\ 0\_b \\ \dots \\ j\_b}} = \begin{cases} \underbrace{\left(2^{q-j}-1\right)}_{\text{Teil } 0\_b} + \underbrace{\left(3\cdot 2^{q-j}-1\right)}_{\text{Teil } 1\_b} - 1 = 2^{q-j+2}-3 = 2^{q+1}-3, & j=1 \\ \underbrace{\left(2^{q-j}-1\right)}_{\text{Teil } 0\_b} + \underbrace{\left(3\cdot 2^{q-j}-1\right)}_{\text{Teil } 1\_b} + \underbrace{\sum_{i=2}^{j}\left(2^i+1\right)\cdot 2^{q-j}}_{\substack{\text{Teile } 2\_b \dots j\_b, \\ \text{mind. 4 Parallelogramme}}} - 1 = \\ \quad = 2^{q+1} + 2^{q-j}\cdot(j-1) - 3, & j \geq 2 \end{cases}$$

(21)

Jetzt kann der XOR-Aufwand des Planes **A1** ermittelt werden, weil alle Summanden von **(17)** in **(18)** und **(21)** ermittelt sind:

$$\#\ XOR^{voll\ iterativ}_{c^{n-GR}_{2n-2}\dots c^{n-GR}_{n}} = \sum_{i=0}^{j-1} \#\ XOR^{Teil}_{i\_a} + \sum_{i=0}^{j-1} \#\ XOR^{iterativ}_{Teil\ i\_b} + \#\ XOR^{voll\ iterativ}_{Teil\ j} = 2^{q+1} - 2^{q-j}(2+j) +$$

$$+ \begin{cases} 2^{q+1}-3, & j=1 \\ 2^{q+1}+2^{q-j}\cdot(j-1)-3, & j\geq 2 \end{cases} = \begin{cases} 2^{q+1}+2^{q-1}-3, & j=1 \\ 2^{q+2}-3\cdot 2^{q-j}-3, & j\geq 2 \end{cases}$$

(22)

Bei der *voll iterativen* Berechnung der linken *n-GR*-Seite beträgt der XOR-Aufwand der Berechnung der Spalte $c^{n\text{-}GR}_{n-1}$ aus dem Spalten-Wert $c^{n\text{-}GR}_{n}$ nur 2 XOR-Gatter: Parallelogramm Nr.1 vom Teil *0_b* und Parallelogramm Nr.2 vom Teil *1_b* verursachen 1 XOR-Gatter je (siehe **Abbildung 9**).

$$\#XOR^{voll\ iterativ}_{c^{n-GR}_{n-1}} = \underbrace{\#XOR^{iterativ}_{c^{\text{alle Teile } i\_a}_{n-1}}}_{=0} + \underbrace{\underbrace{\#XOR^{iterativ}_{c^{\text{alle Teile } i\_b}_{n-1}}}_{=\#XOR_{\substack{\text{iterative Berechnung der Spalten-Differenz} \\ \Delta(c^{\text{alle Teile } i\_b}_{n},\, c^{\text{alle Teile } i\_b}_{n-1})}}} + \underbrace{\#XOR_{\substack{\text{Addition der berechneten} \\ \text{Spalten-Differenz}}}}_{=1}}_{=1} = 2$$

(23)

Nun wird der XOR-Aufwand des Planes **A2** für den Fall *j=q-1* ermittelt. Der einzige Unterschied zu Plan **A1** ist die *separate* Berechnung des Parallelogramms Nr.1 aus **Teil *j***, was aber die *voll iterative* Berechnung des Restes der *n-GR* zur Folge hat. Formel **(24)** zeigt den XOR-Aufwand des Planes **A2**:

$$\#XOR^{A2}_{c^{n-GR}_{2n-2}\dots c^{n-GR}_{n}} = \sum_{i=0}^{j-1} \#\ XOR^{voll\ iterativ}_{i\_a}{}^{Teil} + \sum_{i=0}^{j-1} \#XOR^{iterativ}_{i\_b}{}^{Teil} + \#XOR^{A2}_{Teil\ j} =$$

$$= \sum_{i=0}^{j-1} \#\ XOR^{voll\ iterativ}_{i\_a}{}^{Teil} + \left(\#\ XOR^{voll\ iterativ}_{\substack{0\_b \\ \dots \\ j\_b}}{}^{Teil} - \underbrace{\#\ XOR^{voll\ iterativ}_{\text{Parallelogramm Nr.1 vom Teil j}}}_{=2\cdot 2^{q-j}-1} + \underbrace{\#XOR^{separat}_{\text{Parallelogramm Nr.1 vom Teil j}}}_{=2\cdot 2^{q-j}-3+2^{q-j}-1} \right) =$$

$$= \left(2^{q+1} - 2^{q-j}(2+j) - 1\right) + \left(\begin{cases} 2^{q+1}-3, & q-1=j=1 \Leftrightarrow n=6 \\ 2^{q+1}+2^{q-j}\cdot(j-1)-3, & q-1=j\geq 2 \end{cases} + \left(2^{q-j}-3\right) \right) =$$

$$= \begin{cases} 5, & n=6 \\ 2^{q+2}-11, & q-1=j\geq 2 \end{cases}$$

(24)

Nun wird der XOR-Aufwand des Planes **A3** für *j=q-1* ermittelt. Nach diesem Plan werden alle Parallelogramme des **Teiles *j*** *separat* berechnet. Formel **(25)** zeigt den XOR-Aufwand dieses Ablaufplanes:

$$\#\overset{A3}{XOR}_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}} = \sum_{i=0}^{j-1}\#\overset{voll\ iterativ}{XOR}_{i\_a}^{Teil} + \sum_{i=0}^{j-1}\#\overset{iterativ}{XOR}_{i\_b}^{Teil} + \#\overset{separat}{XOR}_{Teil\ j} = \left(2^{q+1} - 2^{q-j}(2+j) - 1\right) +$$

$$+\begin{cases} \underbrace{\left(2^{q-j}-1\right)}_{\text{Teil } 0\_b} + \underbrace{\underbrace{2\cdot 2^{q-j}-3+2^{q-j}-1}_{\text{Parallelogramm Nr.1}} + \underbrace{2^{q-j}-1}_{\text{Parallelogramm Nr.}2^1}}_{\text{Teil } 1\_b}, & q-1=j=1 \Leftrightarrow n=6 \\ \underbrace{\left(2^{q-j}-1\right)}_{\text{Teil } 0\_b} + \underbrace{\left(3\cdot 2^{q-j}-1\right)}_{\text{Teil } 1\_b} + \left(\begin{array}{l} \underbrace{2\cdot 2^{q-j}-3+2^{q-j}-1}_{\text{Parallelogramm Nr.1}} + \\ + \underbrace{\left(2\cdot 2^{q-j}-1\right)\cdot\left(2^{j}-2\right)}_{\text{Parallelogramme Nr.2}\ldots\text{Nr.}2^{j-1}} + \\ + \underbrace{2^{q-j}-1}_{\text{Parallelogramm Nr.}2^{j}} \end{array}\right), & q-1=j=2 \Leftrightarrow n=14 \\ \underbrace{\left(2^{q-j}-1\right)}_{\text{Teil } 0\_b} + \underbrace{\left(3\cdot 2^{q-j}-1\right)}_{\text{Teil } 1\_b} + \underbrace{\sum_{i=2}^{j-1}\left(2^{i}+1\right)\cdot 2^{q-j}}_{\substack{\text{Teile } 2\_b\ldots(j-1)\_b,\\ \text{mind.4Parallelogramme}}} + \left(\begin{array}{l} \underbrace{2\cdot 2^{q-j}-3+2^{q-j}-1}_{\text{Parallelogramm Nr.1}} + \\ + \underbrace{\left(2\cdot 2^{q-j}-1\right)\cdot\left(2^{j}-2\right)}_{\text{Parallelogramme Nr2}\ldots\text{Nr.}2^{j-1}} + \\ + \underbrace{2^{q-j}-1}_{\text{Parallelogramm Nr.}2^{j}} \end{array}\right), & q-1=j>2 \end{cases} =$$

$$=\begin{cases} 5, & n=6 \\ 22, & n=14 \\ 2^{q+2}+2^{q-1}-14, & q-1=j>2 \end{cases}$$

(25)

Aus dem Vergleich der XOR-Aufwände der Ablaufpläne **A1** (siehe **(22)**), **A2** (siehe **(24)**) und **A3** (siehe **(25)**) folgt, dass für alle *n* mit *j=q-1* der Plan **A2** den minimalen XOR-Aufwand hat. Formel **(26)** zeigt seinen XOR-Aufwand für die Berechnung der Spalte $c^{n\text{-}GR}{}_{n\text{-}1}$ aus dem Spalten-Wert $c^{n\text{-}GR}{}_{n}$:

$$\#\overset{A2}{XOR}_{c_{n-1}^{n-GR}} = \underbrace{\#\overset{iterativ}{XOR}_{c_{n-1}^{alle\ Teile\ i\_a}}}_{=0} + \underbrace{\underbrace{\#\overset{iterativ}{XOR}_{c_{n-1}^{alle\ Teile\ i\_b}}}_{=\#XOR_{\substack{\text{iterative Berechnung der Spalten-Differenz}\\ \Delta(c_n^{alle\ Teile\ i\_b}, c_{n-1}^{alle\ Teile\ i\_b})}}} + \underbrace{\#XOR_{\substack{\text{Addition der berechneten}\\ \text{Spalten-Differenz}}}}_{=1}}_{=1} = 2$$

(26)

Der XOR-Aufwand der Berechnung der *n-GR* nach Plan **A3** im Fall *j=q* kann aus **(25)** unter der Bedingung $\#\overset{separat}{XOR}^{\text{Parallelogramm Nr.1}}_{Teil\ \text{j}} = 0$ folgendermaßen ermittelt werden:

$$\#\overset{A3}{XOR}_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}} = \sum_{i=0}^{j-1}\#\overset{voll\ iterativ}{XOR}^{Teil}_{i\_a} + \sum_{i=0}^{j-1}\#\overset{iterativ}{XOR}^{Teil}_{i\_b} + \#\overset{separat}{XOR}_{Teil\ \text{j}} = \left(2^{q+1} - 2^{q-j}(2+j) - 1\right) +$$

$$+\begin{cases} \underbrace{\left(2^{q-j}-1\right)}_{\text{Teil } 0\_b} + \underbrace{\left(3\cdot 2^{q-j}-1\right)}_{\text{Teil } 1\_b} + \left(\begin{array}{l} \underbrace{0}_{\text{Parallelogramm Nr.1}} + \underbrace{\left(2\cdot 2^{q-j}-1\right)\cdot\left(2^{j}-2\right)}_{\text{Parallelogramme Nr2}\ldots\text{Nr.}2^{j-1}} + \\ + \underbrace{2^{q-j}-1}_{\text{Parallelogramm Nr.}2^{j}} \end{array}\right), & q=j=2 \\ \underbrace{\left(2^{q-j}-1\right)}_{\text{Teil } 0\_b} + \underbrace{\left(3\cdot 2^{q-j}-1\right)}_{\text{Teil } 1\_b} + \underbrace{\sum_{i=2}^{j-1}\left(2^{i}+1\right)\cdot 2^{q-j}}_{\substack{\text{Teile } 2\_b\ldots(j-1)\_b,\\ \text{mind.4Parallelogramme}}} + \left(\begin{array}{l} \underbrace{0}_{\text{Parallelogramm Nr.1}} + \underbrace{2^{q-j}-1}_{\text{Parallelogramm Nr.}2^{j}} \\ + \underbrace{\left(2\cdot 2^{q-j}-1\right)\cdot\left(2^{j}-2\right)}_{\text{Parallelogramme Nr2}\ldots\text{Nr.}2^{j-1}} + \end{array}\right), & q=j>2= \\ 0, \quad q=j=1 \Leftrightarrow n=3 \end{cases}$$

$$=\begin{cases} 0, & q=j=1 \Leftrightarrow n=3 \\ 7, & q=j=2 \Leftrightarrow n=7 \\ 2^{j+2}-9, & q=j>2 \end{cases}$$

(27)

Nun wird der XOR-Aufwand des Planes **A4** für *j=q* ermittelt. Nach diesem Ablaufplan bilden die gefüllten Zellen des **Teils *j*** und des **Teils *(j-1)_b*** die für die Berechnung relevanten Figuren (siehe **Abbildung 10**).

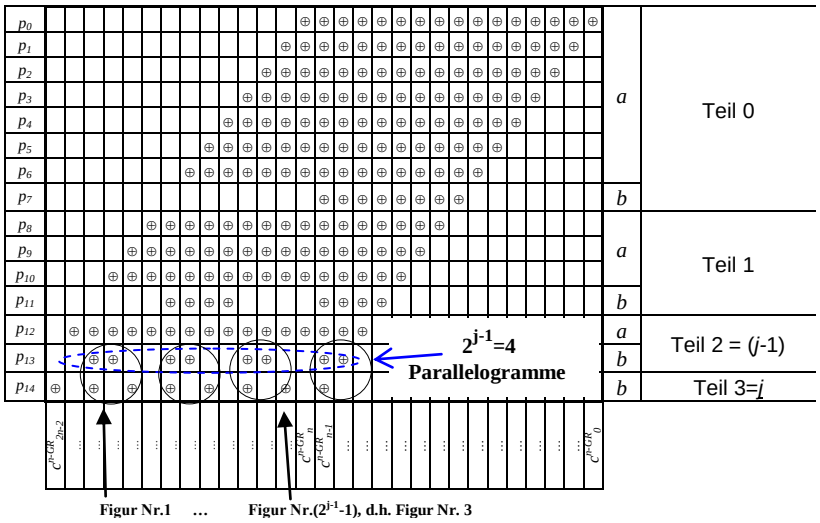


Abbildung 10: *15-GR, n=15=1111₂ ⇒ q=3, j=3*

Die *(2^{j-1}-1)* Figuren werden von Parallelogrammen des **Teils *(j-1)_b*** und von den Parallelogrammen des Teils *j* gebildet. Die Parallelogramme des **Teils *(j-1)_b*** sind nur 1 Zelle hoch und 2 Zellen breit. Die Parallelogramme des **Teils *j*** sind die einzelnen gefüllten Zellen. Jede aus den Parallelogrammen gebildete Figur hat 3 Spalten. Die Berechnung dieser Spalten benötigt nur 1 XOR-Gatter. Für die Addition der Spalten-Werten der Figuren zu dem Rest der *n-GR* sind 3 XOR-Gatter pro Figur ausreichend. Formel **(28)** zeigt der XOR-Aufwand dieses Ablaufplanes.

$$\#\overset{A4}{XOR}_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}} = \sum_{i=0}^{j-1} \#\overset{voll\ iterativ}{XOR}_{i\_a}^{Teil} + \sum_{i=0}^{j-2} \#\overset{iterativ}{XOR}_{i\_b}^{Teil} + \#\overset{A4}{XOR}_{\substack{(j-1)\_b \\ und\ j}}^{Teile} =$$

$$= \left(2^{q+1} - 2^{q-j}\cdot(2+j) - 1\right) +$$

$$+ \begin{cases} 0, & q = j = 1 \Leftrightarrow n = 3 \\ \underbrace{\left(2^{q-j}-1\right)}_{Teil\ 0\_b} + \underbrace{\left(\underbrace{1}_{\substack{Ermittlung\ der \\ Spalten\text{-}Werte \\ der\ Figur}} + \underbrace{3\cdot\left(2^{j-1}-1\right)}_{\substack{Addition(XOR)\ der\ Spalten\text{-}Werte \\ der\ Figuren\ Nr.1\ldots Nr.(2^{j-1}-1)}}\right)}_{Teile\ (j-1)\_b\ und\ j}, & q = j = 2 \Leftrightarrow n = 7 \\ \underbrace{\left(2^{q-j}-1\right)}_{Teil\ 0\_b} + \underbrace{\left(3\cdot 2^{q-j}-1\right)}_{Teil\ 1\_b} + \underbrace{\left(\underbrace{1}_{\substack{Ermittlung\ der \\ Spalten\text{-}Werte \\ der\ Figur}} + \underbrace{3\cdot\left(2^{j-1}-1\right)}_{\substack{Addition(XOR)\ der\ Spalten\text{-}Werte \\ der\ Figuren\ Nr.1\ldots Nr.(2^{j-1}-1)}}\right)}_{Teile\ (j-1)\_b\ und\ j}, & q = j = 3 \Leftrightarrow n = 15 \\ \underbrace{\left(2^{q-j}-1\right)}_{Teil\ 0\_b} + \underbrace{\left(3\cdot 2^{q-j}-1\right)}_{Teil\ 1\_b} + \underbrace{\sum_{i=2}^{j-2}\left(2^i+1\right)\cdot 2^{q-j}}_{\substack{Teile\ 2\_b\ldots(j-2)\_b, \\ mind.4\ Parallelogramme}} + \underbrace{\left(\underbrace{1}_{\substack{Ermittlung\ der \\ Spalten\text{-}Werte \\ der\ Figur}} + \underbrace{3\cdot\left(2^{j-1}-1\right)}_{\substack{Addition(XOR)\ der\ Spalten\text{-}Werte \\ der\ Figuren\ Nr.1\ldots Nr.(2^{j-1}-1)}}\right)}_{Teile\ (j-1)\_b\ und\ j}, & q = j > 3 \end{cases} =$$

$$= \begin{cases} 0, & n = 3 \\ 7, & n = 7 \\ 2^{j+2} - 10, & n \geq 15 \end{cases}$$

(28)

Aus dem Vergleich der XOR-Aufwände der Ablaufpläne **A3** (siehe **(27)**) und **A4** (siehe **(28)**) folgt, dass für alle *n* mit *j=q* der Plan **A4** den minimalen XOR-Aufwand hat. Formel **(29)** zeigt seinen XOR-Aufwand für die Berechnung der Spalte $c^{n\text{-}GR}{}_{n\text{-}1}$ aus dem Spalten-Wert $c^{n\text{-}GR}{}_{n}$ für alle *n>3*. Im Fall *n=3* sind für die Berechnung 2 XOR-Gatter notwendig.

$$\#XOR^{A4}_{c_{n-1}^{n-GR}} = \underbrace{\#XOR^{iterativ}_{c_{n-1}^{Teil\ i\_a}}}_{=0} + \underbrace{\#XOR^{iterativ}_{c_{n-1}^{Teile\ 0\_b\ldots(j-2)\_b}}}_{=\#XOR_{\substack{iterative\ Berechnung\ der\ Spalten\text{-}Differenz \\ \Delta(c_n^{Teile\ 0\_b\ldots(j-2)\_b}, c_{n-1}^{Teile\ 0\_b\ldots(j-2)\_b})}} \underbrace{\#XOR_{\substack{Addition\ der\ berechneten \\ Spalten\text{-}Differenz}}}_{=1} +$$

$$+ \underbrace{\#XOR^{A4}_{\substack{Additionder berechneten \\ Spalten der Figur Nr.2^{j-1}}}}_{=1} = \begin{cases} 2, & j=2 \\ 3, & j>2 \end{cases}$$

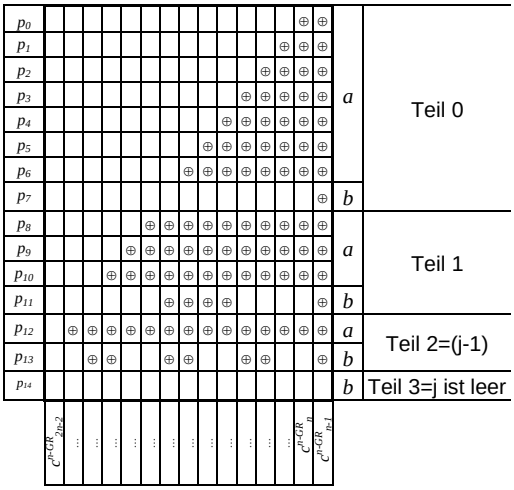
(29)

Tabelle 5 fasst die Ergebnisse kurz zusammen.

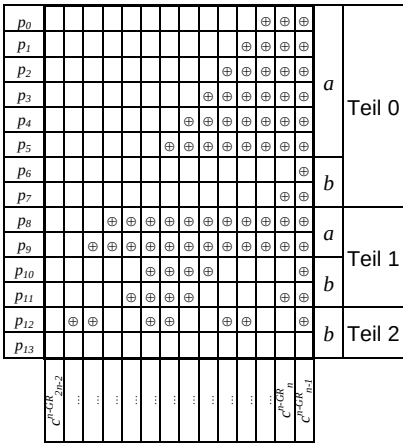
Tabelle 5: Optimale Ablaufpläne und deren XOR-Aufwände für die Berechnung linken Seite n-GR aus Gruppe 3

	Fall	Ablaufplan in **Tabelle 4**	XOR-Aufwand der Berechnung linker n-GR-Seite $\#XOR^{Ablaufplan}_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}}$	XOR-Aufwand der Berechnung der Spalte $c^{n\text{-}GR}_{n\text{-}1}$ $\#XOR^{Ablaufplan}_{c_{n-1}^{n-GR}}$
optimal	$j \le q\text{-}2$	**A1**	siehe *voll iterativen* Ablaufplan	siehe *voll iterativen* Ablaufplan
	$j = q\text{-}1$	**A2**	$\begin{cases} 5 = n-1, & n=6 \\ 2^{q+2}-11 = 2n-7, & n>6 \end{cases}$	2
	$j = q$	**A4**	$\begin{cases} 0 = n-3, & n=3 \\ 7 = n, & n=7 \\ 2^{j+2}-10 = 2n-8, & n \ge 15 \end{cases}$	$\begin{cases} 2, & n \in \{3,7\} \\ 3, & n>7 \end{cases}$
voll iterativ		**A1**	$\begin{cases} 2^{q+1}+2^{q-1}-3 = n+2^q-3, j=1 \\ 2^{q+2}-3\cdot 2^{q-j}-3 = 2n-2^{q-j}-3, j>1 \end{cases}$	2

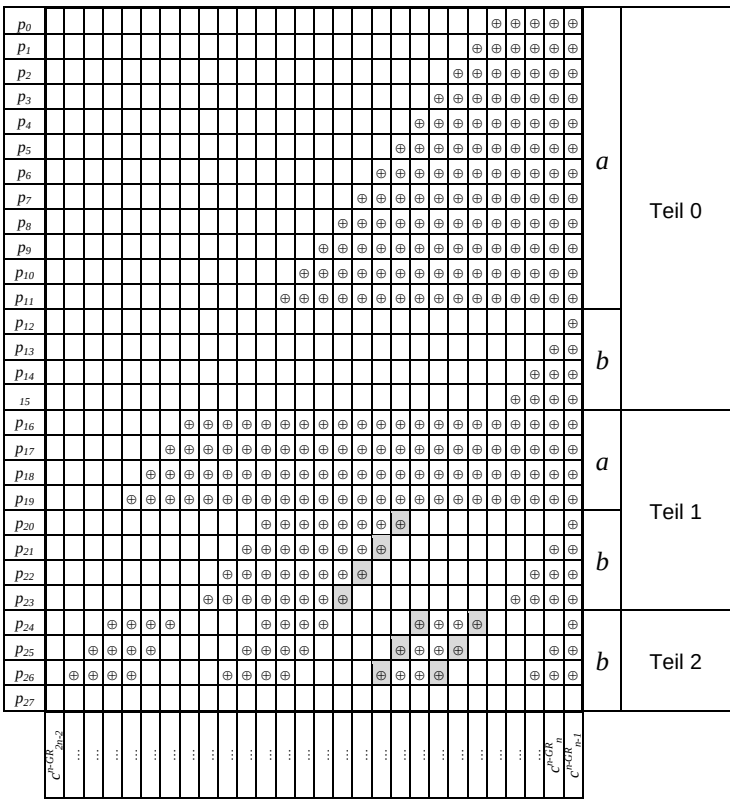
Nun werden die Ablaufpläne der *nicht vollen* Großen Rauten *NV-n*-GR und deren XOR-Aufwände untersucht. Der Unterschied zu den *n*-GR ist nur, dass die Zeile mit dem TP p_{n-1} jetzt leer ist. **Abbildung 11** zeigt die Beispiele der *NV-n-GR* der untersuchten Sonderfälle.



a) NV-15-GR, q=j=3: der Teil *j* ist leer



b) *NV-14-GR, q-1=j=2:* der Teil *j* hat2^j Parallelogramme mit $h_j=2^{q-j}-1=1$ und $L_j=2^{q-j}=2$



c) *NV-28-GR, q=4, j=2, q-j=2*: der Teil j hat 2^j Parallelogramme mit $h_j=2^{q-j}-1$ und $L_j=2^{q-j}$

Abbildung 11: Beispiele der NV-n-GR für die Sonderfälle: *j=q, j=q-1, q-j≥2*

Aus den gleichen Gründen, wie bei n-GR, werden die in **Tabelle 4** aufgelisteten Ablaufpläne untersucht. Bei dem Sonderfall mit *q=j*=1 fehlt der Teil *j* der *NV-n-GR* (siehe **Abbildung 11-a**), so dass sich der XOR-Aufwand nur aus der *voll iterativen* Berechnung der anderen Teile (siehe **(30)**) ergibt.

$$\#\overset{A1}{XOR}_{c_{2n-2}^{NV-n-GR}\ldots c_n^{NV-n-GR}} = \sum_{i=0}^{j-1}\#\overset{voll\ iterativ}{XOR}_{i\_a}^{Teil} + \sum_{i=0}^{j-1}\#XOR_{i\_b}^{iterativ\ Teil} = \left(2^{q+1} - 2^{q-j}(2+j) - 1\right) +$$

$$+\begin{cases} \underbrace{\left(2^{q-j}-1\right)}_{\text{Teil } 0\_b} + \underbrace{\left(3\cdot 2^{q-j}-1\right)}_{\text{Teil } 1\_b}, & q=j=2 \\ \underbrace{\left(2^{q-j}-1\right)}_{\text{Teil } 0\_b} + \underbrace{\left(3\cdot 2^{q-j}-1\right)}_{\text{Teil } 1\_b} + \underbrace{\sum_{i=2}^{j-1}\left(2^i+1\right)\cdot 2^{q-j}}_{\substack{\text{Teile } 2\_b\ldots(j-1)\_b \\ \text{mind. 4 Parallelogramme}}}, & q=j>2 = \begin{cases} 0, & n=3 \\ 2^{j+2}-2^j-7, & q=j\geq 2 \end{cases} \\ 0, & q=j=1 \Leftrightarrow n=3 \end{cases}$$

(30)

Auch im Fall *q-j≥2* (siehe **Abbildung 11-c**) ist der *voll iterative* Ablaufplan optimal. Formel **(31)** zeigt seinen XOR-Aufwand.

$$\#\overset{A1}{XOR}_{c_{2n-2}^{NV-n-GR}\ldots c_n^{NV-n-GR}} = \sum_{i=0}^{j-1}\#XOR_{i\_a}^{iterativ\ Teil} + \sum_{i=0}^{j-1}\#XOR_{i\_b}^{iterativ\ Teil} + \#\overset{voll\ iterativ}{XOR}_{Teil\ j} = 2^{q+1} - 2^{q-j}(2+j) +$$

$$+\begin{cases} \underbrace{\left(2^{q-j}-1\right)}_{\text{Teil } 0\_b} + \underbrace{\left(3\cdot(2^{q-j}-1)-1\right)}_{\text{Teil } 1\_b} - 1, & j=1 \\ \underbrace{\left(2^{q-j}-1\right)}_{\text{Teil } 0\_b} + \underbrace{\left(3\cdot 2^{q-j}-1\right)}_{\text{Teil } 1\_b} + \underbrace{\sum_{i=2}^{j-1}\left(2^i+1\right)\cdot 2^{q-j}}_{\substack{\text{Teile } 2\_b\ldots(j-1)\_b, \\ \text{mind. 4 Parallelogramme}}} + \underbrace{\left(2^j+1\right)\cdot\left(2^{q-j}-1\right)}_{\text{Teil j}} - 1, & j\geq 2 \end{cases} =$$

$$= \begin{cases} 2^{q+1}+2^{q-1}-6, & j=1 \\ 2^{q+2}-3\cdot 2^{q-j}-2^j-4, & j\geq 2 \end{cases}$$

(31)

Die Berechnung der Spalte $c^{NV\text{-}n\text{-}GR}{}_{n\text{-}1}$ aus dem Spalten-Wert $c^{NV\text{-}n\text{-}GR}{}_{n}$ nach dem voll iterativen Ablaufplan benötigt 2 XOR-Gatter für alle *n>3*. Im Fall *n=3* wird nur 1 XOR-Gatter benötigt.

Für den Fall *q-j=1* müssen alle drei Ablaufpläne **A1**, **A2** und **A3** aus der **Tabelle 4** untersucht werden. Unter der Bedingung *q-j=1* kann der XOR-Aufwand des Planes **A1** aus **(31)** folgendermaßen abgeleitet werden:

$$\#\overset{A1}{XOR}_{c_{2n-2}^{NV-n-GR}\ldots c_n^{NV-n-GR}} = \begin{cases} 2^{q+1}+2^{q-1}-6, & q-1=j=1 \\ 2^{q+2}-3\cdot 2^{q-j}-2^j-4, & q-1=j\geq 2 \end{cases} = 2^{q+2}-2^j-10$$

(32)

Formel **(33)** zeigt den XOR-Aufwand des Planes **A2**, nach dem das Parallelogramm Nr.1 vom Teil j *separat* und der Rest der linken *n-GR*-Seite *voll iterativ* berechnet werden:

$$\#\overset{A2}{XOR}_{c_{2n-2}^{NV-n-GR}\ldots c_n^{NV-n-GR}} =$$

$$= \underbrace{\#\overset{A1}{XOR}_{c_{2n-2}^{NV-n-GR}\ldots c_n^{NV-n-GR}} - \underbrace{\#\overset{voll\ iterativ}{XOR}_{\text{vomTeil j}}^{\text{NV-Parallelogramm Nr.1}}}_{=2\cdot(2^{q-j}-1)-1,\,q-j=1} - 1}_{\substack{voll\ iterative \text{ Berechnung der linken } n\text{-}GR\text{-Seite} \\ \text{ohne des nicht vollen NV-Parallelogramm Nr.1 vomTeil j}}} + \underbrace{\#\overset{separat}{XOR}_{\text{vomTeil j}}^{\text{NV-Parallelogramm Nr.1}}}_{=1} =$$

$$= 2^{q+2} - 2^j - 11$$

(33)

Die *separate* Berechnung (siehe Plan **A3**) des **Teils *j*** der *NV-n-GR* hat den folgenden XOR-Aufwand:

$$\#\overset{A3}{XOR}_{c_{2n-2}^{NV-n-GR}\ldots c_n^{NV-n-GR}} = \#\overset{A2}{XOR}_{c_{2n-2}^{NV-n-GR}\ldots c_n^{NV-n-GR}} - \underbrace{\#\overset{iterativ}{XOR}_{\substack{\text{NV-Parallelogramm Nr.2}^j \\ \text{vonTeil j}}}^{\text{NV-Parallelogramm Nr.2}}}_{=\begin{cases}2\cdot(2^{q-j}-1)+(2^j-3)\cdot(2^{q-j}-1),\,q-j=1,n>6\\0,\,n=6\end{cases}} +$$

$$+ \underbrace{\#\overset{separat}{XOR}_{\substack{\text{NV-Parallelogramm Nr.2}^j \\ \text{vonTeil j}}}^{\text{NV-Parallelogramm Nr.2}}}_{=\begin{cases}\left(2^j-2\right)(2\cdot 2^{q-j}-2)+2^{q-j}-2,\,q-j=1,n>6\\0,\,n=6\end{cases}} = \begin{cases}3, n=6\\2^{q+2}-14, n>6\end{cases}$$

(34)

Der Vergleich der Formeln **(32) – (34)** zeigt, dass Plan **A2** der günstigste für die *NV-n-GR*-Berechnung im Fall *q-j=1* ist. Die Berechnung der Spalte $c^{NV\text{-}n\text{-}GR}{}_{n-1}$ aus dem Spalten-Wert $c^{NV\text{-}n\text{-}GR}{}_{n}$ nach diesem Ablaufplan benötigt 2 XOR-Gatter.

Tabelle 6 fasst die Ergebnisse für die *NV-n-GR* kurz zusammen.

Tabelle 6: Optimale Ablaufpläne und deren XOR-Aufwände für die Berechnung linken Seite NV-n-GR aus Gruppe 3

	Fall	Alaufplan in **Tabelle 4**	XOR-Aufwand der Berechnung linker NV-n-GR-Seite $\#\overset{\text{Ablaufplan}}{XOR}_{c_{2n-2}^{NV-n-GR}\ldots c_n^{NV-n-GR}}$	XOR-Aufwand der Berechnung der Spalte $c^{NV\text{-}n\text{-}GR}{}_{n-1}$ $\#\overset{Ablaufplan}{XOR}_{c_{n-1}^{NV-n-GR}}$
optimal	*j≤ q-2*	**A1**	siehe *voll iterativen* Ablaufplan	siehe *voll iterativen* Ablaufplan
	j= q-1	**A2**	$2^{q+2}-2^j-11=2n-2^{q-1}-7$	2
	j= q	**A1**	$\begin{cases}0=n-3, & n=3\\2^{j+2}-2^j-7=2n-2^q-5, & q=j\geq 2\end{cases}$	$\begin{cases}1, & n=3\\2, & n>3\end{cases}$
voll iterativ		**A1**	$\begin{cases}2^{q+1}+2^{q-1}-6=n+2^q-6, & j=1\\2^{q+2}-3\cdot 2^{q-j}-2^j-4=2n-2^{q-j}-2^j-4, & j\geq 2\end{cases}$	2

n-GR-Gruppe 4

Zu dieser Gruppe gehören alle Polynome, deren Länge $n=2^q+2^{q-1}+\ldots+2^{q-j}+r$, mit $j\neq 0$, $r\neq 0 \Rightarrow j\leq q-2$ ist. **Abbildung 12** stellt einen typischen Vertreter der n-GR-Gruppe 4 dar: 27-GR.

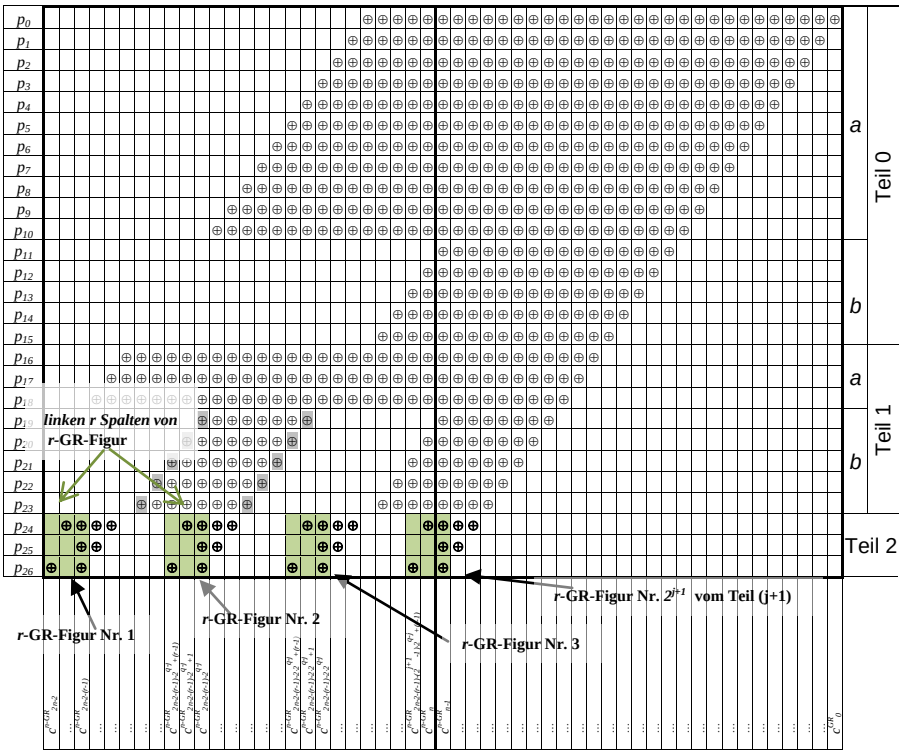


Abbildung 12: Vertreter der n-GR-Gruppe 4, 27-GR,
$n= 27=2^q+2^{q-1}+\ldots+2^{q-j}+r = 11011_2 \Rightarrow q=4,\ j=1$

Die n-GR dieser Gruppe bestehen aus $(j+2)$ Teilen:

• Jeder **Teil** ***i***, $0\leq i\leq j$, besteht aus 2^{q-i} Zeilen und kann folgendermaßen in zwei Sub-Teile aufgeteilt werden:

- **Teil *i_a*** besteht aus $n-\sum_{k=0}^{i} 2^{q-k}$ Zeilen mit $L=2^{q+1}$ ununterbrochen gefüllten Zellen
- **Teil *i_b*** besteht aus $(2^{q-j}-r)$ Zeilen, in denen sich 2^j gleiche Parallelogramme befinden. Die Grundseite jedes Parallelogramms besteht aus $L=2^{q-i}$ ununterbrochen gefüllten Zellen. Die dazugehörige Höhe entspricht $h=2^{q-j}-r$ gefüllten Zellen.

• Der **Teil *(j+1)*** besteht aus *r* Zeilen, die 2^{j+1} *r-GR*-Figuren beinhalten. Die *r-GR*-Figur Nr.1[1] befindet sich in den n-GR-Spalten $c^{n-GR}{}_{2n-2}$ … $c^{n-GR}{}_{2n-2-(2r-1)}$. Die letzte *r-GR*-Figur, d.h. die mit der Nr. 2^{j+1}, befindet sich auf den n-GR-Spalten $c^{n-GR}{}_{n-1-r}$ … $c^{GR}{}_{n-1+(r-1)}$. Ihre mittlere Spalte befindet sich in der mittleren Spalte der ganzen *n*-GR. Der Abstand zwischen allen *r-GR*-Figuren ist gleich $L=2^{q-(j+1)}$. Diese Zahl ist die Anzahl ununterbrochen gefüllter Zellen der Zeile mit dem TP $p_{2^q+\ldots+2^{q-j}+1}= p_{n-r+1}$.

Für alle **Teile *i_a***, $0 \le i \le j$, gilt folgendes: jede Spalte dieses Teils beinhaltet in sich alle TP ihrer linken Nachbar-Spalte und kann ein weiteres Teil-Produkt beinhalten. Analog zu Formel **(121)** (siehe Kapitel 4.3.4.2) ist hier die *iterative* (evtl. *voll iterative*) Berechnung optimal. Auf alle anderen n-GR-Teile mit Ausnahme der *r-GR*-Figur Nr.1 von **Teil *(j+1)*** sind jetzt die Kriterien (***K***) anwendbar.
Aus allen n-GR-Teilen hat der **Teil *j*** die kleinste Anzahl ununterbrochen gefüllter Zellen: $L_{0-b)} > L_{1-b)} > \ldots > L_{(j-1)-b)} > L_{j-b)} = 2^{q-j}$, wobei $min(2^{q-j}) = 2^{q-(q-2)} = 4$. Nach den Kriterien ***(K)*** gehören alle **Teile *i_b***, $0 \le i \le j$, zu dem *iterativen* TD-Teil.

Für alle Parallelogramme jedes **Teils *i_b***, $1 \le i \le j$, gilt folgendes: Die linken Seiten aller Parallelogramme ab Nr. 2 befinden sich direkt unter den Seiten der Parallelogramme des **Teils *(i-1)_b*** (siehe **Abbildung 12**). Dasselbe gilt auch für alle *r-GR*-Figuren ab *r-GR*-Figur Nr.2: Die linken *(r-1)* Spalten dieser Figuren befinden sich in den gleichen *n-GR*-Spalten wie auch die schrägen Seiten der Parallelogramme aus dem **Teil *j_b*** (siehe **Abbildung 12**). Diese Position der Parallelogramme und der *r-GR*-Figuren begünstigt die *iterative* Berechnung der ganzen linken *n-GR*-Seite. **Abbildung 13** zeigt einen Teil der 27-GR, um das zu illustrieren. Zuerst werden die Summen (XOR) der Spalten-Differenzen der linken Seite des Parallelogramms Nr.1 vom **Teil *j_b*** zusammen mit den jeweiligen Spalten-Differenzen der *r-GR*-Figur Nr.2 ermittelt. Die erste Verwendung dieser bereits ermittelten Werte (in **Abbildung 13** grau gekennzeichnet) findet bei der Berechnung der GR-Spalten, die die rechte Seite des Parallelogramms Nr.1 vom Teil *j* beinhalten, statt. Dieselben Werte können für die Berechnung der Spalten-Differenzen der GR-Spalten, die die linke Seite des Parallelogramms Nr.1 vom Teil *(j-1)_b* beinhalten, wieder verwendet werden. Hierdurch wird die *iterative* Berechnung der ganzen n-GR zusätzlich begünstigt.

[1] Alle Figuren wurden von links nach rechts nummeriert.

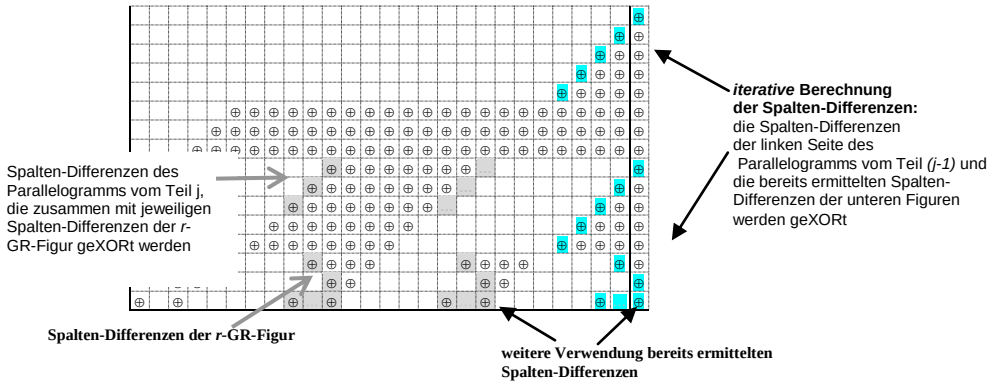


Abbildung 13: Die Position der Parallelogramme und der *r-GR*-Figuren begünstigt die *iterative* Berechnung

Andererseits sind die linken *r* Spalten der *r-GR*-Figur Nr.1 der einzige Inhalt der linken *r* Spalten der ganzen GR, was die *separate* Berechnung des **Teils *(j+1)*** begünstigen kann. Folgendes muss aber berücksichtigt werden: einerseits ist die *separate* Berechnung der linken Seite der *r-GR*-Figur Nr.1 zu bevorzugen, anderseits ist die gleiche Art der Berechnung der gleichen *r-GR*-Figur-Seiten zu bevorzugen, eben die *iterative* Berechnung linker Seiten aller anderen *r-GR*-Figuren. Aus diesem Grund müssen für alle *n* mit *r>1* die folgenden Fälle untersucht werden:

- Wenn die ganze *r-GR*-Figur Nr.1 *separat* und die linken Seiten der anderen *r-GR*-Figuren *iterativ* berechnet werden, wie viele Spalten-Differenzen müssen dann – für *r* oder für *(r-1)* linke Spalten – ermittelt werden?
- Ist die *iterative* Berechnung der linken Seiten aller *r-GR*-Figur einschließlich Nr.1 die günstigste?

Im Fall *r=1* gehört der Teil (*j+1*) eindeutig zum *separaten* TD-Teil (siehe Kriterien (***K***)).

Um die Berechnung des XOR-Aufwandes mehrerer Ablaufpläne im Fall *r>1* zu vermeiden, kann zuerst die folgende Frage beantwortet werden: Sollen die linken Seiten aller *r-GR*-Figur *separat* berechnet werden, wenn die linke Seite der *r-GR*-Figur Nr.1 *separat* berechnet wurde, oder ist die *iterative* Berechnung in diesem Fall optimal?
Um diese Frage zu beantworten, müssen die folgenden XOR-Aufwände verglichen werden: der XOR-Aufwand der *separaten* Berechnung aller linken *r-GR*-Figur-Seiten und der XOR-Aufwand der *separaten* Berechnung der linken Seiten nur *der r-GR*-Figur Nr.1 und der *iterativen* Berechnung der anderen Figuren. In beiden

Fällen werden die linken r Spalten der *r-GR*-Figur Nr.1 *separat* mit $\#\overset{optimal}{XOR}_{c_{2r-2}^{r-GR},\ldots,c_{r-1}^{r-GR}}$ XOR-Gattern berechnet. Der Teil (*j+1*) beinhaltet 2^{j+1} *r-GR*-Figuren. Die *separate* Berechnung der linken *r* Spalten jeder *r-GR*-Figur ab Nr.2 bis Nr.(2^{j+1}-*1*) benötigt *r* XOR-Gatter pro Figur unter der Bedingung, dass ihre Spalten-Werte bereits zur Verfügung stehen. Die Addition (XOR) der (*r-1*) Spalten der *r-GR*-Figur Nr.2^{j+1} benötigt (*r-1*) XOR-Gatter.

$$\#\overset{separat}{XOR}_{\substack{linke\ Seiten\ aller\\ r-GR-Figuren}} = \#\overset{optimal}{XOR}_{c_{2r-2}^{r-GR},\ldots,c_{r-1}^{r-GR}} + \sum_{i=2}^{2^{j+1}-1} \#XOR^{\substack{Addition\\ der\ linken\ r\\ Spalten-Werte\ der\\ r-GR-Figur Nr.i}} + \#XOR^{\substack{Addition\\ der\ linken\\ r-1\\ Spalten-Werte\ der\\ r-GR-Figur Nr.2^{j+1}}} =$$

$$= \#\overset{optimal}{XOR}_{c_{2r-2}^{r-GR},\ldots,c_{r-1}^{r-GR}} + \left(2^{j+1}-2\right)\cdot r + (r-1) = \#\overset{optimal}{XOR}_{c_{2r-2}^{r-GR},\ldots,c_{r-1}^{r-GR}} + \left(2^{j+1}-1\right)\cdot r - 1$$

(35)

Formel **(36)** zeigt den XOR-Aufwand der *iterativen* Berechnung der linken Seiten dieser Figuren. Wegen der Lage der Figuren und der Parallelogramme des **Teils *j*** besteht der XOR-Aufwand dieser Berechnung aus *iterativer* Berechnung linker Seite nur einer *r-GR*-Figur.

$$\#\overset{optimal}{XOR}_{c_{2r-2}^{r-GR},\ldots,c_{r-1}^{r-GR}} + \#\overset{iterativ}{XOR}^{\substack{linken\\ Seiten\\ r-GR-Figur\\ Nr.2\ldots Nr.2^{j+1}}} = \#\overset{optimal}{XOR}_{c_{2r-2}^{r-GR},\ldots,c_{r-1}^{r-GR}} + \sum_{i=0}^{r-1} \#\overset{iterativ}{XOR}_{\Delta\left(c_{2r-2-i+1}^{r-GR},c_{2r-2-i}^{r-GR}\right)} +$$

$$+ \underbrace{\#XOR^{\substack{Additionder\\ berechneten\\ Spalten-Differenzen\\ mit den jeweiligen\\ Spalten-Differenzen\\ des Teils j}}}_{=r} = \#\overset{optimal}{XOR}_{c_{2r-2}^{r-GR},\ldots,c_{r-1}^{r-GR}} + \sum_{i=0}^{r-1}\left(\#\overset{iterativ}{XOR}_{\Delta\left(c_{2r-2-i+1}^{r-GR},c_{2r-2-i}^{r-GR}\right)} + 1\right) =$$

$$= \#\overset{optimal}{XOR}_{c_{2r-2}^{r-GR},\ldots,c_{r-1}^{r-GR}} + \#\overset{iterativ}{XOR}_{c_{2r-2}^{r-GR},\ldots c_{r-1}^{r-GR}}$$

(36)

Aus dem Vergleich der Formeln **(35)** und **(36)** ergibt sich folgendes: die *iterative* Berechnung wird bevorzugt, wenn die Ungleichung **(37)** wahr ist.

$$\#\overset{iterativ}{XOR}^{\text{linke } r \text{ Spalten}}_{r-GR-Figur} < \left(2^{j+1}-1\right)\cdot r - 1$$

(37)

Das Maximum der linken Seite der Ungleichung **(37)** [2] ist nicht größer als 2*r*. Die rechte Seite der Ungleichung **(37)** erreicht ihr Minimum *3r-1*, wenn *j=1*. Daraus folgt:

$$\#\overset{iterativ}{XOR}^{\text{linke } r \text{ Spalten}}_{r-GR-Figur} \le 2r < 3r-1 < \left(2^{j+1}-1\right)\cdot r - 1$$

---

[2] $\#\overset{iterativ}{XOR}^{\text{linke } r \text{ Spalten}}_{r-GR-Figur} = 1 + \#\overset{voll\ iterativ}{XOR}^{\text{linke } r \text{ Spalten}}_{r-GR-Figur} = \begin{cases} r, & r-\text{GR aus Gruppe 1} \\ 2r, & r-\text{GR aus Gruppe 2} \\ 2r-1, & r-\text{GR aus Gruppe 3} \end{cases}$

Damit ist die Ungleichung **(37)** für alle *r>1* immer wahr, was die *iterative* Berechnung der linken Seiten aller *r-GR*-Figuren ab Nr.2 optimal macht.

Die Fälle, die zu untersuchen bleiben, beziehen sich auf die verschieden Berechnungs-Arten der *r-GR*-Figur Nr.1. Im Fall der *voll iterativer* Berechnung der beiden Seiten der *r-GR*-Figur Nr.1 – genauso wie im Fall der *separaten* Berechnung ihrer linker Seite mit *iterativer* Berechnung ihrer rechter Seite – ist die *iterative* Berechnung der beiden Seiten anderer *r-GR*-Figuren optimal. Wenn die ganze *r-GR*-Figur Nr.1 *separat* berechnet wird, können entweder beide Seiten der anderen *r-GR*-Figuren *iterativ* berechnet werden, oder nur deren linke *r* (oder (*r-1*)) Spalten *iterativ* und die anderen Spalten werden *separat* berechnet. Die Sonderfälle, die untersucht werden müssen, sind in **Tabelle 7** zusammengefasst.

Tabelle 7: Untersuchte Ablaufpläne für n-GR-Gruppe 4

<table>
<tr><td rowspan="3">Fall</td><td>Teile 0…j</td><td colspan="8">Teil (j+1)</td><td rowspan="3">Bezeichnung des Ablaufplanes</td></tr>
<tr><td></td><td colspan="3">r-GR-Figur Nr.1</td><td colspan="2">r-GR-Figur Nr.2</td><td>r-GR-Figur Nr.3</td><td>…</td><td>r-GR-Figur Nr.2^j</td></tr>
<tr><td></td><td>linke Seite</td><td colspan="2">rechte Seite</td><td>linke Seite</td><td>rechte Seite</td><td>linke Seite</td><td>rechte Seite</td><td></td><td>linke Seite, (r-1) Spalten</td></tr>
<tr><td rowspan="5">r>1</td><td rowspan="2">iterativ</td><td>voll iterativ</td><td>iterativ</td><td>iterativ</td><td>iterativ</td><td>iterativ</td><td colspan="2">siehe r-GR-Figur Nr.2</td><td>…</td><td>siehe r-GR-Figur Nr.2</td><td>A1</td></tr>
<tr><td>separat</td><td>separat</td><td>iterativ</td><td>iterativ</td><td>iterativ</td><td colspan="2">siehe r-GR-Figur Nr.2</td><td>…</td><td>siehe r-GR-Figur Nr.2</td><td>A2</td></tr>
<tr><td rowspan="3">voll iterativ</td><td colspan="3" rowspan="3">separat</td><td colspan="2">iterativ</td><td colspan="2">siehe r-GR-Figur Nr.2</td><td>…</td><td>siehe r-GR-Figur Nr.2</td><td>A3-a</td></tr>
<tr><td>iterativ, r Spalten</td><td>separat, (r-1) Spalten</td><td colspan="2">siehe r-GR-Figur Nr.2</td><td>…</td><td>siehe r-GR-Figur Nr.2</td><td>A3-b</td></tr>
<tr><td>iterativ, (r-1) Spalten</td><td>separat, r Spalten</td><td colspan="2">siehe r-GR-Figur Nr.2</td><td>…</td><td>siehe r-GR-Figur Nr.2</td><td>A3-c</td></tr>
<tr><td>r=1</td><td>voll iterativ</td><td colspan="3">separat</td><td colspan="2">separat</td><td>separat</td><td>…</td><td>separat</td><td>A4</td></tr>
</table>

Es wird mit der *voll iterativen* Berechnung der linken *n-GR*-Seite begonnen, was Plan **A1** (siehe **Tabelle 7**) entspricht. Der XOR-Aufwand dieses Ablaufplans kann folgendermaßen berechnet werden:

$$\#\ XOR_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}}^{voll\ iterativ} = \sum_{i=0}^{j} \#\ XOR_{i\_a}^{iterativ,\ Teil} + \sum_{i=0}^{j} \#\ XOR_{i\_b}^{iterativ,\ Teil} + \#\ XOR_{(j+1)}^{voll\ iterativ,\ Teil} \qquad \textbf{(38)}$$

Jetzt wird jeder Summand aus **(38)** einzeln ermittelt. Formel **(39)** zeigt den XOR-Aufwand der *iterativen* Berechnung aller **Teile *i_a***.

$$\sum_{i=0}^{j} \#\,XOR_{i\_a}^{Teil\ iterativ} = \sum_{i=0}^{j} \left(2^{q-i} - 2^{q-j} + r\right) = 2^{q+1} - 2^{q-j}(2+j) + (j+1)\cdot r =$$
$$= 2^{q+1} - 2^{q-j+1} + j\cdot\left(r - 2^{q-j}\right) + r$$

(39)

Wegen der Position der Parallelogramme aller **Teile *i_b*** (siehe **Abbildung 13**) besteht die Möglichkeit, gewisse Spalten-Differenzen *iterativ* zu berechnen. Diese Möglichkeit gilt nur für die linken Seiten der Parallelogramme. Diese Tatsache kann auch folgendermaßen umformuliert werden: Die zwei linken Parallelogramme jedes **Teils *i_b*** mit $2 \le i \le j$ tragen mit ihren beiden Seiten zum XOR-Aufwand bei, und jedes weiteres Parallelogramm trägt nur mit seiner rechten Seite bei. Formel **(40)** zeigt den XOR-Aufwand der *iterativen* Berechnung aller Teile *i_b*.

$$\sum_{i=0}^{j} \#\,XOR_{i\_b}^{Teil\ iterativ} = \#\,XOR_{0\_b}^{Teil\ iterativ} + \#\,XOR_{1\_b}^{Teil\ iterativ} + \sum_{i=2}^{j} \#\,XOR_{i\_b}^{Teil\ iterativ} = \underbrace{\left(2^{q-j} - r - 1\right)}_{\substack{\text{Teil 0\_b, nur 1 Parallelogramm,}\\ \text{linke } \left(2^{q-j}-r-1\right)\text{Spalten}}} +$$

$$+ \underbrace{\left(\underbrace{2\cdot\left(2^{q-j}-r\right)}_{\text{Parallelogramm Nr.1}} + \underbrace{2^{q-j}-r-1}_{\substack{\text{linke}\left(2^{q-j}-r-1\right)\text{Spalten}\\ \text{des Parallelogramms Nr.2}}} \right)}_{\substack{\text{Teil 1\_b,}\\ \text{2 Parallelogramme}}} + \underbrace{\sum_{i=2}^{j} \left(\underbrace{2\cdot\left(2^{q-j}-r\right)}_{\text{Parallelogramm Nr.1}} + \underbrace{2\cdot\left(2^{q-j}-r\right)}_{\text{Parallelogramm Nr.2}} + \underbrace{\left(2^{i}-3\right)\cdot\left(2^{q-j}-r\right)}_{\substack{\text{rechte Seiten anderer}\\ \text{Parallelogramme}}} \right)}_{\substack{\text{Teile 2\_b...j\_b,}\\ \text{mind.4 Parallelogramme}}} =$$

$$= \begin{cases} 2^{q-j+2} - 4\cdot r - 2, & j = 1 \\ 2^{q-j+2} - 4\cdot r - 2 + \left(2^{q-j} - r\right)\cdot \sum_{i=2}^{j}\left(2^{i}+1\right), & j > 1 \end{cases}$$

(40)

Die *iterativ*e Berechnung der Teile *0* bis *j* benötigt die folgende Anzahl XOR-Gatter:

$$\#\,XOR_{0\ldots j}^{Teile\ iterativ} = \sum_{i=0}^{j} \#\,XOR_{i\_a}^{Teil\ iterativ} + \sum_{i=0}^{j} \#\,XOR_{i\_b}^{Teil\ iterativ} = 2^{q+1} - 2^{q-j+1} + j\cdot\left(r - 2^{q-j}\right) + r +$$
$$+ \begin{cases} 2^{q-j+2} - 4\cdot r - 2, & j = 1 \\ 2^{q-j+2} - 4\cdot r - 2 + \left(2^{q-j} - r\right)\cdot \sum_{i=2}^{j}\left(2^{i}+1\right), & j > 1 \end{cases} = 2^{q+2} - 3\cdot 2^{q-j} - 2\cdot r\cdot\left(2^{j}-1\right) - 2$$

(41)

Die *voll iterative* Berechnung dieser Teile, die bei den Plänen **A3-a**, **A3-b**, **A3-c** und **A4** vorgesehen ist, benötigt wegen der *separaten* Berechnung der Start-Spalte ein XOR-Gatter weniger:

$$\#\,XOR_{0\ldots j}^{Teile\ voll\ iterativ} = \#\,XOR_{0\ldots j}^{Teile\ iterativ} - 1 = 2^{q+2} - 3\cdot 2^{q-j} - 2\cdot r\cdot\left(2^{j}-1\right) - 3$$

(42)

Jetzt kann der XOR-Aufwand des Planes **A1** ermittelt werden:

$$\#\overset{voll\ iterativ}{XOR}_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}} = \#\overset{A1}{XOR}_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}} = \sum_{i=0}^{j}\#\overset{iterativ}{XOR}^{Teil}_{i\_a} + \sum_{i=0}^{j}\#\overset{iterativ}{XOR}^{Teil}_{i\_b} + \#\overset{voll\ iterativ}{XOR}_{(j+1)}^{Teil} =$$

$$= \#\overset{iterativ}{XOR}^{Teile}_{0\ldots j} + \#\overset{voll\ iterativ}{XOR}_{(j+1)}^{Teil} = \left(2^{q+2} - 3\cdot 2^{q-j} - 2\cdot r\cdot\left(2^j - 1\right) - 2\right) +$$

$$+ \left(\underbrace{\underbrace{\#\overset{voll\ iterativ}{XOR}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#\overset{voll\ iterativ}{XOR}_{c_{r-1}^{r-GR}}}_{\text{linke } r \text{ Spalten}} + \underbrace{r}_{\substack{\text{rechte}\\ \text{Seite}}}}_{r-GR-\text{Figur Nr.1}} + \underbrace{\underbrace{2\cdot r}_{r-GR-\text{Figur Nr.2}} + \underbrace{\left(2^{j+1}-3\right)\cdot r}_{\substack{r-GR-\text{Figur}\\ \text{Nr.3}\ldots\text{Nr.}\left(2^{j+1}-1\right)}}}_{} \right) =$$

$$= 2^{q+2} - 3\cdot 2^{q-j} + 2\cdot\left(r-1\right) + \#\overset{voll\ iterativ}{XOR}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#\overset{voll\ iterativ}{XOR}_{c_{r-1}^{r-GR}}$$

(43)

Unter Berücksichtigung der Tatsache, dass $n=2^q+2^{q-1}+\ldots+2^{q-j}+r=2^{q+1}-2^{q-j}+r$, kann Formel **(43)** folgendermaßen umgewandelt werden:

$$\#\overset{voll\ iterativ}{XOR}_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}} = \#\overset{A1}{XOR}_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}} = 2^{q+2} - 3\cdot 2^{q-j} + 2\cdot\left(r-1\right) + \#\overset{voll\ iterativ}{XOR}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} +$$

$$+ \#\overset{voll\ iterativ}{XOR}_{c_{r-1}^{r-GR}} = \underbrace{2^{q+2} - 2\cdot 2^{q-j} + 2\cdot r}_{=2n} - 2^{q-j} - 2 + \#\overset{voll\ iterativ}{XOR}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#\overset{voll\ iterativ}{XOR}_{c_{r-1}^{r-GR}} =$$

$$= 2n - 2^{q-j} - 2 + \#\overset{voll\ iterativ}{XOR}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#\overset{voll\ iterativ}{XOR}_{c_{r-1}^{r-GR}}$$

(44)

Bei der *voll iterativen* Berechnung der linken *n-GR*-Seite beträgt der XOR-Aufwand der Berechnung der Spalte c^{n-GR}_{n-1} aus dem Spalten-Wert c^{n-GR}_{n} nur 2 XOR-Gatter, die die Spalten der linken Seite des Parallelogramms Nr.1 von **Teil *0_b*** und des Parallelogramms Nr.2 von **Teil *1_b*** verursachen:

$$\#\overset{voll\ iterativ}{XOR}_{c_{n-1}^{n-GR}} = \underbrace{\#\overset{iterativ}{XOR}_{c_{n-1}^{\text{alle Teile } i\_a}}}_{=0} + \underbrace{\underbrace{\#\overset{iterativ}{XOR}_{c_{n-1}^{Teile\ \ i\_b..(j+1)}}}_{=\#XOR_{\substack{\text{iterative Berechnung der Spalten-Differenz}\\ \Delta(c_n^{Teile\ \ i\_b..j},\, c_{n-1}^{Teile\ \ i\_b..(j+1)})}}}_{=1} + \underbrace{\#XOR_{\substack{\text{Addition der berechneten}\\ \text{Spalten-Differenz}}}}_{=1}}_{} = 2$$

(45)

Nun wird der XOR-Aufwand des Planes **A2** ermittelt. Der Unterschied zu Plan **A1** ist die *separate* Berechnung der linken *r* Spalten der *r-GR*-Figur Nr.1. Formel **(46)** zeigt den XOR-Aufwand des Planes **A2**:

$$\#\overset{A2}{XOR}_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}} = \#\overset{iterativ}{XOR}^{Teile}_{0\ldots j} + \#\overset{A2}{XOR}^{Teil}_{(j+1)} = \left(2^{q+2} - 3\cdot 2^{q-j} - 2\cdot r\cdot\left(2^j - 1\right) - 2\right) +$$

$$+ \left(\begin{array}{l} \underbrace{\#\overset{optimal}{XOR}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#\overset{optimal}{XOR}_{c_{r-1}^{r-GR}}}_{\text{linke Seite der } r-GR-FigurNr.1} + \underbrace{r}_{\substack{\text{rechte}\\ \text{Seite}}} + \\ + \underbrace{1 + \#\overset{voll\ iterativ}{XOR}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#\overset{voll\ iterativ}{XOR}_{c_{r-1}^{r-GR}}}_{\text{linke } r \text{ Spalten}} + \underbrace{r}_{\substack{\text{rechte}\\ \text{Seite}}} + \underbrace{\left(2^{j+1} - 3\right)\cdot r}_{\substack{r-GR-\text{Figur}\\ \text{Nr.3}\ldots\text{Nr.}\left(2^{j+1}-1\right)}} \\ \underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{r-GR-\text{FigurNr.2}} \end{array} \right) =$$

$$= \#\overset{voll\ iterativ}{XOR}_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}} + \left(\#\overset{optimal}{XOR}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#\overset{optimal}{XOR}_{c_{r-1}^{r-GR}} \right) - r + 1 =$$

$$= \underbrace{2^{q+2} - 3\cdot 2^{q-j} + r - 1}_{2^{q+2} - 2\cdot 2^{q-j} + 2\cdot r - 1 - 2^{q-j} - r = 2n - 2^{q-j} - r - 1} +$$

$$+ \left(\#\overset{voll\ iterativ}{XOR}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#\overset{voll\ iterativ}{XOR}_{c_{r-1}^{r-GR}} \right) + \left(\#\overset{optimal}{XOR}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#\overset{optimal}{XOR}_{c_{r-1}^{r-GR}} \right) =$$

$$= 2n - 2^{q-j} - r - 1 + \#\overset{voll\ iterativ}{XOR}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#\overset{voll\ iterativ}{XOR}_{c_{r-1}^{r-GR}} + \#\overset{optimal}{XOR}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#\overset{optimal}{XOR}_{c_{r-1}^{r-GR}}$$

(46)

Aus dem Vergleich von **(46)** und **(44)** folgt, dass der Plan **A1** bevorzugt werden soll, weil die Ungleichung **(47)** immer wahr[3] ist.

$$\#\overset{optimal}{XOR}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#\overset{optimal}{XOR}_{c_{r-1}^{r-GR}} \geq r - 1$$

(47)

Falls *r>1* eine ungerade Zahl ist, beinhaltet nicht nur die Spalte $c^{n\text{-}GR}_{2n\text{-}2}$, sondern auch die Spalte $c^{n\text{-}GR}_{2n\text{-}3}$ jeweils nur ein TP. In dem Fall kann die *separate* Berechnung der Spalten $c^{n\text{-}GR}_{2n\text{-}2}$ und $c^{n\text{-}GR}_{2n\text{-}3}$ in Kombination mit der *iterativen* Berechnung des Restes der GR (weiter bezeichnet als Plan **A1-b**) die folgende Verringerung des XOR-Aufwandes bewirken:

$$\#\overset{A1-b}{XOR}_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}} = \#\overset{iterativ}{XOR}^{Teile}_{0\ldots j} + \#\overset{A1-b}{XOR}^{Teil}_{(j+1)} =$$

$$= \#\overset{voll\ iterativ}{XOR}_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}} - \underbrace{\#XOR_{\substack{\text{Addition der}\\ \text{Spalten-Differenz}\\ \Delta\left(c_{2r-2}^{r-GR}, c_{2r-3}^{r-GR}\right)\\ \text{zum Wert } c_{2n-2}^{n-GR}}}}_{=1} + \underbrace{\#\overset{separat}{XOR}_{\substack{\text{Start-Spalte}\\ c_{2n-3}^{n-GR}}}}_{=0} = \#\overset{voll\ iterativ}{XOR}_{c_{2n-2}^{n-GR} - c_n^{n-GR}} - 1$$

(48)

Nun werden die Ablaufpläne **A3-a**, **A3-b** und **A3-c** verglichen. Die Formeln **(49) – (51)** zeigen deren XOR-Aufwände.

---

[3] Der XOR-Aufwand der Berechnung der *r* verschiedenen, nicht leeren, Spalten kann nicht kleiner als *r-1* XOR-Gatter werden, weil in dem Fall, wenn eine der Spalten nur 1 Teil-Produkt beinhaltet, der XOR-Aufwand der Berechnung dieser Spalte 0 Gatter beträgt. Alle anderen (*r-1*) verschiedenen, nicht leeren, Spalten benötigen mindestens 1 XOR-Gatter pro Spalte.

$$\#XOR^{A3-a}_{c_{2n-2}^{n-GR}..c_n^{n-GR}} = \#XOR^{voll\ iterativ}_{0...j}{}^{Teile} + \#XOR^{A3-a}_{(j+1)}{}^{Teil} = \left(2^{q+2} - 3\cdot 2^{q-j} - 2\cdot r\cdot\left(2^j - 1\right) - 3\right) +$$

$$+\left(\begin{array}{l} \underbrace{\underbrace{\#XOR^{optimal}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}}}_{\text{linke } r \text{ Spalten}} + \underbrace{r-1}_{\substack{\text{rechte}\\ \text{Seite}}} + \underbrace{r-1}_{\substack{\text{Addition der rechten } r\text{-1 Spalten}\\ \text{mit den jeweiligen}\\ \text{Spalten-Werten}\\ \text{der Teile 0...j}}} + }_{r-GR-\text{Figur Nr.1}} \\ \underbrace{+1+\#XOR^{voll\ iterativ}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#XOR^{voll\ iterativ}_{c_{r-1}^{r-GR}} + \underbrace{r-1}_{\substack{\text{rechte}\\ r-1\\ \text{Spalten}}} + \underbrace{1}_{\substack{\text{Addition des}\\ \text{Wertes } c_0^{r-GR}}} + \underbrace{\left(2^{j+1}-3\right)\cdot r}_{\substack{r-GR-\text{Figur}\\ \text{Nr.3...Nr.}\left(2^{j+1}-1\right)}}}_{r-GR-\text{Figur Nr.2}} \end{array}\right) =$$

$$= 2^{q+2} - 3\cdot 2^{q-j} + 2\cdot r + \#XOR^{voll\ iterativ}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#XOR^{voll\ iterativ}_{c_{r-1}^{r-GR}} + \#XOR^{optimal}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} - 4 =$$

$$= 2^{q+2} - 3\cdot 2^{q-j} + 2\cdot r + \#XOR^{voll\ iterativ}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#XOR^{voll\ iterativ}_{c_{r-1}^{r-GR}} + \#XOR^{optimal}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} - 4$$

$$= \underbrace{2^{q+2} - 3\cdot 2^{q-j} + 2\cdot r}_{=2n-2^{q-j}} - 4 + \#XOR^{optimal}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#XOR^{voll\ iterativ}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#XOR^{voll\ iterativ}_{c_{r-1}^{r-GR}} =$$

$$= 2n - 2^{q-j} - 4 + \#XOR^{optimal}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#XOR^{voll\ iterativ}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#XOR^{voll\ iterativ}_{c_{r-1}^{r-GR}} =$$

$$= \#XOR^{A1}_{c_{2n-2}^{n-GR}..c_n^{n-GR}} + \#XOR^{optimal}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} - 2$$

(49)

$$\#XOR^{A3-b}_{c_{2n-2}^{n-GR}..c_n^{n-GR}} = \#XOR^{voll\ iterativ}_{0...j}{}^{Teile} + \#XOR^{A3-b}_{(j+1)}{}^{Teil} = \left(2^{q+2} - 3\cdot 2^{q-j} - 2\cdot r\cdot\left(2^j - 1\right) - 3\right) +$$

$$+\left(\begin{array}{l} \underbrace{\underbrace{\#XOR^{optimal}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}}}_{\text{linke } r \text{ Spalten}} + \underbrace{r-1}_{\substack{\text{rechte}\\ \text{Seite}}} + \underbrace{r-1}_{\substack{\text{Addition der rechten } r\text{-1 Spalten}\\ \text{mit den jeweiligen}\\ \text{Spalten-}\\ \text{Werten}\\ \text{der Teile 0...j}}} }_{r-GR-\text{Figur Nr.1}} \\ \underbrace{+1+\#XOR^{voll\ iterativ}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#XOR^{voll\ iterativ}_{c_{r-1}^{r-GR}} + \underbrace{1}_{\substack{\text{Addition}\\ \text{des}\\ \text{Wertes}\\ c_{r-1}^{r-GR}}} + \underbrace{r-1}_{\substack{\text{rechte}\\ r-1\\ \text{Spalten}}} + }_{\text{linken } r \text{ Spalten}} \\ \underbrace{+\left(2^{j+1}-3\right)\cdot \underbrace{1}_{\substack{\text{Addition}\\ \text{des Wertes}\\ c_{r-1}^{r-GR}}} + \left(2^{j+1}-3\right)\cdot(r-1)}_{r-GR-\text{Figuren Nr.3...Nr.}\left(2^{j+1}-1\right)} \end{array}\right) =$$

$$= \underbrace{2^{q+2} - 3\cdot 2^{q-j} + 2\cdot r}_{=2n-2^{q-j}} - 4 + \#XOR^{optimal}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#XOR^{voll\ iterativ}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#XOR^{voll\ iterativ}_{c_{r-1}^{r-GR}} =$$

$$= 2n - 2^{q-j} - 4 + \#XOR^{optimal}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#XOR^{voll\ iterativ}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#XOR^{voll\ iterativ}_{c_{r-1}^{r-GR}}$$

$$= \#XOR^{A1}_{c_{2n-2}^{n-GR}..c_n^{n-GR}} + \#XOR^{optimal}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} - 2$$

(50)

$$\#XOR^{A3-c}_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}} = \#XOR^{voll\ iterativ}_{0\ldots j}{}^{Teile} + \#XOR^{A3-c}_{(j+1)}{}^{Teil} = \left(2^{q+2} - 3\cdot 2^{q-j} - 2\cdot r\cdot\left(2^j - 1\right) - 3\right) +$$

$$+ \left(\underbrace{\underbrace{\#XOR^{optimal}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}}}_{\text{linke } r \text{ Spalten}} + \underbrace{r-1}_{\substack{\text{rechte}\\ \text{Seite}}} + \underbrace{r-1}_{\substack{\text{Addition der rechten } r\text{-1 Spalten mit jeweiligen Spalten-Werten der Teile } 0\ldots j}}}_{r-GR-\text{Figur Nr.1}} + \right.$$

$$\left. + \underbrace{\underbrace{1 + \#XOR^{voll\ iterativ}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}}}_{\text{linke } r \text{ Spalten}} + \underbrace{1}_{\substack{\text{Addition}\\ \text{des Wertes}\\ c_r^{r-GR}}} + \underbrace{r}_{\substack{\text{rechte}\\ r \text{ Spalten}}}}_{r-GR-\text{Figur Nr.2}} + \underbrace{\left(2^{j+1}-3\right)\cdot r}_{\substack{r-GR-\text{Figur}\\ \text{Nr.3}\ldots\text{Nr.}\left(2^{j+1}-1\right)}} \right) = \underbrace{2^{q+2} - 3\cdot 2^{q-j} + 2r}_{=2n-2^{q-j}} - 3 +$$

$$+ \#XOR^{optimal}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} + \#XOR^{voll\ iterativ}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} = 2n - 2^{q-j} - 3 + \#XOR^{optimal}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} +$$

$$+ \#XOR^{voll\ iterativ}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} = \#XOR^{A1}_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}} + \#XOR^{optimal}_{c_{2r-2}^{r-GR}\ldots c_r^{r-GR}} - \left(\#XOR^{voll\ iterativ}_{c_{r-1}^{r-GR}} + 1\right)$$

(51)

Aus dem Vergleich von **(49)** und **(50)** folgt, dass die Pläne **A3-a** und **A3-b** gleichaufwendig sind.

Unter der Berücksichtigung der Tatsache, dass $\#XOR^{voll\ iterativ}_{c_{r-1}^{r-GR}} \geq 1$ ist, folgt aus dem Vergleich **(49) – (51)**, dass die Pläne **A3-a** und **A3-b** nie besser als der Plan **A3-c** sind.

Der XOR-Aufwand der Berechnung des Wertes $c^{n\text{-}GR}_{n\text{-}1}$ aus dem Wert $c^{n\text{-}GR}_{n}$ nach Plan **A3-c** beträgt insgesamt 3 XOR-Gatter: das Parallelogramm Nr.1 des Teils *0_b*, das Parallelogramm Nr.2 des Teils *1_b* und die Addition (XOR) des Wertes $c^{r\text{-}GR}_{r\text{-}1}$ tragen mit je 1 XOR-Gatter bei:

$$\#XOR^{voll\ iterativ}_{c_{n-1}^{n-GR}} = \underbrace{\#XOR^{iterativ}_{c_{n-1}^{i\_a}}{}^{\text{alle Teile}}}_{=0} + \underbrace{\underbrace{\#XOR^{iterativ}_{c_{n-1}^{i\_b}}{}^{\text{alle Teile}}}_{=\#XOR_{\substack{\text{iterative}\\ \text{Berechnung}\\ \text{der}\\ \text{Spalten-Differenz}}}\left(c_n^{i\_b}{}^{\text{alle Teile}}, c_{n-1}^{i\_b}{}^{\text{alle Teile}}\right)} + \underbrace{\#XOR_{\substack{\text{Addition der}\\ \text{berechneten}\\ \text{Spalten-}\\ \text{Differenzen}}}}_{=1}}_{=1} + \underbrace{\underbrace{\#XOR^{separat}_{c_{n-1}^{Teil\ (j+1)}}}_{=\#XOR^{optimal}_{c_{r-1}^{r-GR}}} + \underbrace{\#XOR_{\substack{\text{Addition des}\\ \text{Wertes } c_{r-1}^{r-GR}}}}_{=1}}_{=0,\text{bereits berechnet}} = 3$$

(52)

Aus dem Vergleich des Planes **A1** für gerade *r* (siehe **(44)**) und Planes **A1-b** für ungerade *r* (siehe **(48)**) mit dem Plan **A3-c** (siehe **(51)**) folgt, dass der Plan **A3-c** nur dann besser ist, wenn die Ungleichung **(53)** stimmt.

$$\#XOR^{optimal}_{c^{r-GR}_{2r-2}\ldots c^{r-GR}_{r}} - \left(\#XOR^{voll\ iterativ}_{c^{r-GR}_{r-1}} + 1\right) < 0, \quad \text{gerades} \quad r$$

$$\#XOR^{optimal}_{c^{r-GR}_{2r-2}\ldots c^{r-GR}_{r}} - \#XOR^{voll\ iterativ}_{c^{r-GR}_{r-1}} < 0, \quad \text{ungerades} \quad r$$

(53)

Der XOR-Aufwand der Berechnung des Wertes $c^{r\text{-}GR}_{r\text{-}1}$ nach dem *voll iterativen* Ablaufplan beträgt maximal ein XOR-Gatter für die GR-Gruppe 1 und zwei XOR-Gatter für die GR-Gruppen 2, 3 und 4, d.h. $\max\left(\#XOR^{voll\ iterativ}_{c^{r-GR}_{r-1}}\right) = 2$ in **(53)**.

Ungleichung **(53)** ist nur für *r≤3* wahr. Aus diesem Grund ist der Plan **A3-c** nur für $r \in \{2,3\}$ optimal. Für diese Werte von *r* kann **(51)** folgendermaßen umgewandelt werden:

$$\#XOR^{A3-c}_{c^{n-GR}_{2n-2}\ldots c^{n-GR}_{n}} = 2n - 2^{q-j} - 3 + \underbrace{\#XOR^{optimal}_{c^{r-GR}_{2r-2}\ldots c^{r-GR}_{r}}}_{=0, r\in\{2,3\}} + \underbrace{\#XOR^{voll\ iterativ}_{c^{r-GR}_{2r-2}\ldots c^{r-GR}_{r}}}_{=\begin{cases}0, r=2\\ 2, r=3\end{cases}} =$$

$$= 2n - 2^{q-j} - 3 + \begin{cases}0, r = 2\\ 2, r = 3\end{cases}$$

(54)

Formel **(55)** zeigt den XOR-Aufwand für den Sonderfall *r=1*. In diesem Fall ist die *voll iterative* Berechnung der Teile *0* bis *j* mit der *separaten* Berechnung des Teils (*j+1*) optimal (siehe Plan **A4** in **Tabelle 7**).

$$\#XOR^{A4}_{c^{n-GR}_{2n-2}\ldots c^{n-GR}_{n}} = \#XOR^{voll\ iterativ}_{Teile\ 0\ldots j} + \#XOR^{separat}_{Teil\ (j+1)} =$$

$$= \left(2^{q+2} - 3\cdot 2^{q-j} - 2\cdot \underbrace{r}_{=1}\cdot\left(2^{j}-1\right) - 3\right) + \left(2^{j+1} - 2\right) = 2^{q+2} - 3\cdot 2^{q-j} - 3 = 2n - 2^{q-j} - 5$$

(55)

Für den XOR-Aufwand der Berechnung des Wertes $c^{n\text{-}GR}_{n\text{-}1}$ aus dem Wert $c^{n\text{-}GR}_{n}$ nach Plan **A4** gilt Formel **(52)**.

Für den XOR-Aufwand der *voll iterativen* Berechnung der ganzen linken *n-GR*-Seite im Fall *r=1* gilt Formel **(44)** unter der Bedingung $\#XOR^{voll\ iterativ}_{c^{r-GR}_{2r-2}\ldots c^{r-GR}_{r}} = 0,\ \#XOR^{voll\ iterativ}_{c^{r-GR}_{r-1}} = 0$.

Tabelle 8 fasst die Ergebnisse für n-GR-Gruppe 4 kurz zusammen.

Tabelle 8: Optimale Ablaufpläne und deren XOR-Aufwände für die Berechnung linken Seite n-GR aus Gruppe 4

	Fall		Ablaufplan in **Tabelle 7**	XOR-Aufwand der Berechnung linker n-GR-Seite Ablaufplan $\#\ XOR_{c_{2n-2}^{n-GR}..c_n^{n-GR}}$	XOR-Aufwand der Berechnung der Spalten $c^{n\text{-}GR}_{n-1}$ *Ablaufplan* $\#\ XOR_{c_{n-1}^{n-GR}}$
optimal	*r=1*		**A4**	$2n-2^{q-j}-5$	3
	$r \in \{2,3\}$		**A3-c**	$2n-2^{q-j}-3+\begin{cases}0, & r=2\\ 2, & r=3\end{cases}$	3
	r≥4	*gerade*	**A1**	siehe *voll iterativ*	2
		ungerade	**A1-b**	$2n-2^{q-j}-3+\overset{voll\ iterativ}{\#\ XOR}_{c_{2r-2}^{r-GR}..c_r^{r-GR}}+\overset{voll\ iterativ}{\#\ XOR}_{c_{r-1}^{r-GR}}$	2
voll iterativ	*r=1*		**A1**	$2n-2^{q-j}-2$	2
	r>1		**A1**	$2n-2^{q-j}-2+\overset{voll\ iterativ}{\#\ XOR}_{c_{2r-2}^{r-GR}..c_r^{r-GR}}+\overset{voll\ iterativ}{\#\ XOR}_{c_{r-1}^{r-GR}}$	2

Nun werden die Ablaufpläne und deren XOR- Aufwände der *nicht vollen* Großen Rauten *NV-n*-GR untersucht.

Aus den gleichen Gründen, wie bei n-GR, werden die in **Tabelle 9** aufgelisteten Ablaufpläne untersucht. Der Unterschied zu *n*-GR ist nur, dass die Zeile mit dem TP p_{n-1} jetzt leer ist und der Teil (*j+1*) jetzt 2^{j+1} *NV-r-GR*-Figuren (*nicht vollen r-GR*-Figuren) beinhaltet. Die linke Seite der *NV-r-GR*-Figur hat nur *r-2* Spalten, d.h. im Vergleich zu der *r-GR*-Figur eine Spalte weniger. Außerdem ist der Inhalt der Spalten $c^{NV\text{-}r\text{-}GR}{}_{r-1}$ und $c^{NV\text{-}r\text{-}GR}{}_{r-2}$ gleich. Deswegen ist im Fall der *separaten* Berechnung der ganzen *NV-r-GR*-Figur Nr.1 die *iterative* Berechnung der linken *r* Spalten, einschließlich der Spalte $c^{NV\text{-}r\text{-}GR}{}_{r-2}$, aller *NV-r-GR*-Figuren ab Nr.2 zu bevorzugen (siehe Plan **A3-b** in **Tabelle 9**).
Auch die Berechnung der *NV-n-GR* unterscheidet sich von Plan **A4** in **Tabelle 7**, wenn *r=1* gilt. In diesem Fall ist der ganze Teil (*j+1*) leer, und die Teile 0…j werden *voll iterativ* berechnet, was Plan **A1** entspricht. Für den XOR-Aufwand ist Formel **(42)** unter der Bedingung *r=1* gültig. Für den XOR-Aufwand der Berechnung der Spalte $c^{NV\text{-}n\text{-}GR}{}_{n-1}$ aus dem Spalten-Wert $c^{NV\text{-}n\text{-}GR}{}_{n}$ gilt in diesem Fall Formel **(45)**.

Tabelle 9: Untersuchte Ablaufpläne für NV-n-GR aus Gruppe 4

Fall	Teile *0…j*	Teil *(j+1)*								Bezeichnung des Ablaufplanes
		NV-r-GR-Figur Nr.1			*NV-r-GR*-Figur Nr.2		*NV-r-GR*-Figur Nr.3	…	*NV-r-GR*-Figur Nr.2^j	
		linke Seite, (*r-2*) Spalten	rechte Seite: 1 Spalte	rechte Seite: *r-1* Spalten	linke Seite	rechte Seite	linke Seite	rechte Seite	linke Seite, *(r-2) Spalten*	
r>1	*iterativ*	*voll iterativ*	*iterativ*	*iterativ*	*iterativ*	*iterativ*	siehe *NV-r-GR*-Figur Nr.2	…	siehe *NV-r-GR*-Figur Nr.2	**A1**
		separat	***separat***	*iterativ*	*iterativ*	*iterativ*	siehe *NV-r-GR*-Figur Nr.2	…	siehe *NV-r-GR*-Figur Nr.2	**A2**
	voll iterativ	***separat***			*iterativ*		siehe *NV-r-GR*-Figur Nr.2	…	siehe *NV-r-GR*-Figur Nr.2	**A3-a**
					iterativ, *r* Spalten	***separat***, *(r-2)* Spalten	siehe *r-GR*-Figur Nr.2	…	siehe *NV-r-GR*-Figur Nr.2	**A3-b**
r=1	*voll iterativ*	-			-		-	…	-	**A1**

Die XOR-Aufwände der *NV-n*-GR können aus den Formeln der *n-GR* unter Berücksichtigung folgender Tatsachen ermittelt werden:

- Anstatt des XOR-Aufwandes der Berechnung der linken (*r-1*) (oder *r*) Spalten der *r-GR*-Figur wird nun der XOR-Aufwand der Berechnung der linken (*r-1*) (oder *r*) Spalten der nicht vollen *NV-r-GR*-Figur benötigt
- Der XOR-Aufwand der Berechnung der rechten Seite der nicht vollen *NV-r-GR*-Figur beträgt *(r-1)* XOR-Gatter, d.h. im Vergleich zu der *r-GR*-Figur ein XOR-Gatter weniger.

Formel **(56)** zeigt den XOR-Aufwand der *voll iterativen* Berechnung der linken *n-GR*-Seite (siehe Plan **A1** in **Tabelle 9**) und Formel **(57)** zeigt den XOR-Aufwand des Planes **A2**.

$$\#\overset{voll\ iterativ}{XOR}_{c_{2n-3}^{NV-n-GR}\ldots c_n^{NV-n-GR}} = \#\overset{A1}{XOR}_{c_{2n-2}^{NV-n-GR}\ldots c_n^{NV-n-GR}} = \#\overset{iterativ}{XOR}^{Teile}_{0\ldots j} + \#\overset{voll\ iterativ}{XOR}_{(j+1)}^{Teil} =$$

$$= \left(2^{q+2} - 3\cdot 2^{q-j} - 2\cdot r\cdot\left(2^j - 1\right) - 2\right) + \left(\begin{array}{l} \underbrace{\underbrace{\#\overset{voll\ iterativ}{XOR}_{c_{2r-3}^{NV-r-GR}\ldots c_r^{NV-r-GR}} + \#\overset{voll\ iterativ}{XOR}_{c_{r-1}^{NV-r-GR}}}_{\text{linker–1Spalten}} + \underbrace{r-1}_{\text{rechte Seite}} +}_{NV-r-GR-\text{FigurNr.1}} \\ \\ + \underbrace{2\cdot(r-1)}_{\substack{NV-r-GR-\\ \text{FigurNr.2}}} + \underbrace{\left(2^{j+1}-3\right)\cdot(r-1)}_{\substack{NV-r-GR-\text{Figur}\\ \text{Nr.3…Nr.}\left(2^{j+1}-1\right)}} \end{array} \right) =$$

$$= 2^{q+2} - 3\cdot 2^{q-j} + 2\cdot(r-1) + \#\overset{voll\ iterativ}{XOR}_{c_{2r-3}^{NV-r-GR}\ldots c_r^{NV-r-GR}} + \#\overset{voll\ iterativ}{XOR}_{c_{r-1}^{NV-r-GR}} - 2^{j+1} =$$

$$= 2n - 2^{q-j} - 2^{j+1} - 2 + \#\overset{voll\ iterativ}{XOR}_{c_{2r-3}^{NV-r-GR}\ldots c_r^{NV-r-GR}} + \#\overset{voll\ iterativ}{XOR}_{c_{r-1}^{NV-r-GR}}$$

(56)

$$\#\overset{A2}{XOR}_{c_{2n-3}^{NV-n-GR}\ldots c_n^{NV-n-GR}} = \#\overset{iterativ}{XOR}^{Teile}_{0\ldots j} + \#\overset{A2}{XOR}^{Teil}_{(j+1)} = \left(2^{q+2} - 3\cdot 2^{q-j} - 2\cdot r\cdot\left(2^j - 1\right) - 2\right) +$$

$$+\left(\begin{array}{l} \underbrace{\#\overset{optimal}{XOR}_{c_{2r-3}^{NV-r-GR}\ldots c_r^{NV-r-GR}} + \#\overset{optimal}{XOR}_{c_{r-1}^{NV-r-GR}}}_{\text{linke Seite der NV-}r\text{-}GR\text{-}Figur Nr.1} + \underbrace{r-1}_{\substack{\text{rechte}\\ \text{Seite}}} + \\ \\ \underbrace{\underbrace{1 + \#\overset{voll\ iterativ}{XOR}_{c_{2r-3}^{NV-r-GR}\ldots c_r^{NV-r-GR}} + \#\overset{voll\ iterativ}{XOR}_{c_{r-1}^{NV-r-GR}}}_{\text{linke } r-1 \text{ Spalten}} + \underbrace{r-1}_{\substack{\text{rechte}\\ \text{Seite}}}}_{NV-r-GR-\text{FigurNr.2}} + \underbrace{\left(2^{j+1}-3\right)\cdot(r-1)}_{\substack{NV-r-GR-\text{Figur}\\ \text{Nr.3}\ldots\text{Nr.}\left(2^{j+1}-1\right)}} \end{array}\right) =$$

$$= \#\overset{voll\ iterativ}{XOR}_{c_{2n-3}^{NV-n-GR}\ldots c_n^{NV-n-GR}} + \left(\#\overset{optimal}{XOR}_{c_{2r-3}^{NV-r-GR}\ldots c_r^{NV-r-GR}} + \#\overset{optimal}{XOR}_{c_{r-1}^{NV-r-GR}}\right) - r + 2 =$$

$$= 2^{q+2} - 3\cdot 2^{q-j} - 2^{j+1} + r + \left(\#\overset{voll\ iterativ}{XOR}_{c_{2r-3}^{NV-r-GR}\ldots c_r^{NV-r-GR}} + \#\overset{voll\ iterativ}{XOR}_{c_{r-1}^{NV-r-GR}}\right) +$$

$$+\left(\#\overset{optimal}{XOR}_{c_{2r-3}^{NV-r-GR}\ldots c_r^{NV-r-GR}} + \#\overset{optimal}{XOR}_{c_{r-1}^{NV-r-GR}}\right)$$

(57)

Für alle *r>1* ist der Plan **A1** günstiger als der Plan **A2**, weil die Bedingung **(58)** immer wahr ist.

$$\underbrace{\#\overset{optimal}{XOR}_{c_{2r-3}^{NV-r-GR}\ldots c_r^{NV-r-GR}} + \#\overset{optimal}{XOR}_{c_{r-1}^{NV-r-GR}}}_{\text{optimale Berechnung der } (r\text{-}1) \text{ verschiedenen Spalten}} \quad \geq \quad r-2$$

(58)

Der XOR-Aufwand der Berechnung des Wertes $c^{NV\text{-}n\text{-}GR}_{n-1}$ aus dem Wert $c^{NV\text{-}n\text{-}GR}_{n}$ nach Plan **A1** beträgt 2 XOR-Gatter (siehe Formel **(45)**, die auch hier gültig ist).

Die Formeln **(59)** und **(60)** zeigen den XOR-Aufwand der Pläne **A3-a** und **A3-b**.

$$\#\overset{A3-a}{XOR}_{c_{2n-3}^{NV-n-GR}\ldots c_n^{NV-n-GR}} = \#\overset{voll\ iterativ}{XOR}^{Teile}_{0\ldots j} + \#\overset{A3-a}{XOR}^{Teil}_{(j+1)} = \left(2^{q+2} - 3\cdot 2^{q-j} - 2\cdot r\cdot\left(2^j - 1\right) - 3\right) +$$

$$+\left(\begin{array}{l} \underbrace{\underbrace{\#\overset{optimal}{XOR}_{c_{2r-3}^{NV-r-GR}\ldots c_r^{NV-r-GR}}}_{\text{linke } r-1 \text{ Spalten}} + \underbrace{r-2}_{\substack{\text{rechte}\\ \text{Seite}}} + \underbrace{r-1}_{\substack{\text{Addition der rechten } r\text{-1 Spalten mit den jeweiligen}\\ \text{Spalten-Werten der Teile 0...j}}} +}_{NV-r-GR-\text{FigurNr.1}} \\ \\ \underbrace{\underbrace{1 + \#\overset{voll\ iterativ}{XOR}_{c_{2r-3}^{NV-r-GR}\ldots c_r^{NV-r-GR}} + \#\overset{voll\ iterativ}{XOR}_{c_{r-1}^{NV-r-GR}}}_{\text{linke } r-1 \text{ Spalten}} + \underbrace{r-2}_{\substack{\text{rechte}\\ r-1\\ \text{Spalten}}} + \underbrace{1}_{\substack{\text{Addition}\\ \text{des Wertes}\\ c_0^{NV-r-GR}}}}_{NV-r-GR-\text{FigurNr.2}} + \underbrace{\left(2^{j+1}-3\right)\cdot(r-1)}_{\substack{NV-r-GR-\text{Figur}\\ \text{Nr.3}\ldots\text{Nr.}\left(2^{j+1}-1\right)}} \end{array}\right) =$$

$$= 2^{q+2} - 3 \cdot 2^{q-j} + 2 \cdot r - 2^{j+1} - 3 + \#XOR^{optimal}_{c_{2r-3}^{NV-r-GR} \ldots c_r^{NV-r-GR}} + \#XOR^{voll\ iterativ}_{c_{2r-3}^{NV-r-GR} \ldots c_r^{NV-r-GR}} +$$
$$+ \#XOR^{voll\ iterativ}_{c_{r-1}^{NV-r-GR}} =$$
$$= 2n - 2^{q-j} - 2^{j+1} - 3 + \#XOR^{optimal}_{c_{2r-3}^{NV-r-GR} \ldots c_r^{NV-r-GR}} + \#XOR^{voll\ iterativ}_{c_{2r-3}^{NV-r-GR} \ldots c_r^{NV-r-GR}} + \#XOR^{voll\ iterativ}_{c_{r-1}^{NV-r-GR}} =$$
$$= \#XOR^{A1}_{c_{2n-3}^{NV-n-GR} \ldots c_n^{NV-n-GR}} + \#XOR^{optimal}_{c_{2r-3}^{NV-r-GR} \ldots c_r^{NV-r-GR}} - 1$$

(59)

$$\#XOR^{A3-b}_{c_{2n-3}^{NV-n-GR} \ldots c_n^{NV-n-GR}} = \#XOR^{voll\ iterativ}_{0 \ldots j}{}^{Teile} + \#XOR^{A3-b}_{(j+1)}{}^{Teil} = \left(2^{q+2} - 3 \cdot 2^{q-j} - 2 \cdot r \cdot \left(2^j - 1\right) - 3\right) +$$
$$+ \underbrace{\underbrace{\#XOR^{optimal}_{c_{2r-3}^{NV-r-GR} \ldots c_r^{NV-r-GR}}}_{\text{linke } r-1 \text{ Spalten}} + \underbrace{r-2}_{\substack{\text{rechte} \\ \text{Seite}}} + \underbrace{r-1}_{\substack{\text{Addition der rechten } r\text{-1 Spalten mit den jeweiligen} \\ \text{Spalten-Werten der Teile } 0 \ldots j}} +}_{NV-r-GR-\text{Figur Nr.1}}$$

$$+ \underbrace{\underbrace{1 + \#XOR^{voll\ iterativ}_{c_{2r-3}^{NV-r-GR} \ldots c_r^{NV-r-GR}} + \#XOR^{voll\ iterativ}_{c_{r-1}^{NV-r-GR}} + \underbrace{\#XOR^{voll\ iterativ}_{c_{r-2}^{NV-r-GR}}}_{=0} + \underbrace{1}_{\substack{\text{Addition} \\ \text{des Wertes} \\ c_{r-2}^{NV-r-GR}}}}_{\text{linke } r \text{ Spalten}} + \underbrace{r-2}_{\substack{\text{rechte} \\ r-2 \text{ Spalten,} \\ \text{separat}}} +}_{NV-r-GR-\text{Figur Nr.2}}$$

$$+ \underbrace{\left(2^{j+1} - 3\right) \cdot \underbrace{1}_{\substack{\text{Addition} \\ \text{des Wertes} \\ c_{r-2}^{NV-r-GR}}} + \left(2^{j+1} - 3\right) \cdot (r-2)}_{\substack{NV-r-GR-\text{Figur} \\ \text{Nr.3} \ldots \text{Nr.} \left(2^{j+1}-1\right)}} = 2^{q+2} - 3 \cdot 2^{q-j} - 2^{j+1} + 2 \cdot r - 3 +$$
$$+ \#XOR^{optimal}_{c_{2r-3}^{NV-r-GR} \ldots c_r^{NV-r-GR}} + \#XOR^{voll\ iterativ}_{c_{2r-3}^{NV-r-GR} \ldots c_r^{NV-r-GR}} + \#XOR^{voll\ iterativ}_{c_{r-1}^{NV-r-GR}} = 2n - 2^{q-j} - 2^{j+1} - 2 +$$
$$+ \#XOR^{voll\ iterativ}_{c_{2r-3}^{NV-r-GR} \ldots c_r^{NV-r-GR}} + \#XOR^{voll\ iterativ}_{c_{r-1}^{NV-r-GR}} + \#XOR^{optimal}_{c_{2r-3}^{NV-r-GR} \ldots c_r^{NV-r-GR}} - 1 =$$

$$= \#XOR^{A1}_{c_{2n-3}^{NV-n-GR} \ldots c_n^{NV-n-GR}} + \#XOR^{optimal}_{c_{2r-3}^{NV-r-GR} \ldots c_r^{NV-r-GR}} - 1$$

(60)

Aus dem Vergleich von **(59)** und **(60)** folgt, dass die Pläne **A3-b** und **A3-c** gleichaufwändig sind. Aus dem Vergleich dieser Pläne mit dem Plan **A1** (siehe **(56)**) folgt, dass sie unter der Bedingung **(61)** günstiger sind als Plan **A1**.

$$\#XOR^{optimal}_{c_{2r-3}^{NV-r-GR} \ldots c_r^{NV-r-GR}} - 1 < 0$$

(61)

Bedingung **(61)** ist nur für *1<r≤3* wahr. Für diese Werte von *r* kann **(59)** folgendermaßen umgewandelt werden:

$$\#^{A3-a}XOR_{c_{2n-3}^{NV-n-GR}\ldots c_n^{NV-n-GR}} = 2n-2^{q-j}-2^{j+1}-3+\underbrace{\#^{optimal}XOR_{c_{2r-3}^{NV-r-GR}\ldots c_r^{NV-r-GR}}}_{=0,r\in\{2,3\}}+$$

$$+\underbrace{\#^{voll\ iterativ}XOR_{c_{2r-3}^{NV-r-GR}\ldots c_r^{NV-r-GR}}+\#^{voll\ iterativ}XOR_{c_{r-1}^{NV-r-GR}}}_{=\begin{cases}0,r=2\\1,r=3\end{cases}}=2n-2^{q-j}-2^{j+1}-3+\begin{cases}0,r=2\\1,r=3\end{cases} \quad \textbf{(62)}$$

Der XOR-Aufwand der Berechnung des Wertes $c^{NV\text{-}n\text{-}GR}{}_{n-1}$ aus dem Wert $c^{NV\text{-}n\text{-}GR}{}_{n}$ nach Plan **A3-a** beträgt 2 XOR-Gatter (siehe **(45)**, die auch hier gültig ist).

Die XOR-Aufwände der Berechnung der NV-n-GR sind in **Tabelle 10** zusammen gefasst.

Tabelle 10: Optimale Ablaufpläne und deren XOR-Aufwände für die Berechnung linken Seite NV-n-GR aus Gruppe 4

	Fall	Ablaufplan in **Tabelle 9**	XOR-Aufwand der Berechnung linker n-GR-Seite $\#^{Ablaufplan}XOR_{c_{2n-3}^{NV-n-GR}\ldots c_n^{NV-n-GR}}$	XOR-Aufwand der Berechnung der Spalten $c^{n\text{-}GR}{}_{n-1}$ $\#^{Ablaufplan}XOR_{c_{n-1}^{NV-n-GR}}$
optimal	*r=1*	**A1**	siehe *voll iterativ*	2
	$r\in\{2,3\}$	**A3-a**	$2n-2^{q-j}-2^{j+1}-3+\begin{cases}0, & r=2\\1, & r=3\end{cases}$	
	r≥4	**A1**	siehe *voll iterativ*	
voll iterativ	*r=1*	**A1**	$2^{q+2}-3\cdot 2^{q-j}-2^{j+1}-1=2n-2^{q-j}-2^{j+1}-3$	
	r>1	**A1**	$2n-2^{q-j}-2^{j+1}-2+\#^{voll\ iterativ}XOR_{c_{2r-3}^{NV-r-GR}\ldots c_r^{NV-r-GR}}+\#^{voll\ iterativ}XOR_{c_{r-1}^{NV-r-GR}}$	

Anhang 2

Herleitung des XOR-Aufwandes der Winograd-*n*-GR

Hier wird der XOR-Aufwand der optimalen Berechnung der linken Seite der Winograd-n-GR (siehe Formel **(145)** in Kapitel 4.4.2) hergeleitet. In Kapitel 4.4.2 wurde gezeigt (siehe Formel **(143)**), dass der XOR-Aufwand folgendermaßen beschrieben werden kann:

$$\#\overset{optimal}{XOR}_{c_{2n-2}^{n-GR}\ldots c_n^{n-GR}} = \#\overset{iterativ}{XOR}_{\text{Teil }0} + \#\overset{optimal}{XOR}_{\text{Teil }i_{\min}} +$$
$$+\begin{cases} 0, & i_{\min}=1 \\ 2\cdot N_{1\_a} + \left(2\cdot K_{1\_b}+3\right)\cdot N_{1\_b} - 1 + \begin{cases} -N_{1\_b}, & i_{\min}>1,\ n\text{-GR aus Gruppe } 2 + \\ 0, & i_{\min}>1,\ n\text{-GR aus Gruppe } 3 \end{cases} \end{cases}$$
$$+\begin{cases} 0, & i_{\min}=1 \\ \sum_{j=2}^{i_{\min}-1}\left(2\cdot N_{j\_a} + \left(2\cdot K_{j\_b}+3\right)\cdot N_{j\_b}\right), & i_{\min}>2 \end{cases}$$

mit

$$\#\overset{iterativ}{XOR}_{\text{Teil }0} = \begin{cases} 3^i - 1, & n\text{-GR aus Gruppe } 2 \\ 2\cdot 3^i - 1, & n\text{-GR aus Gruppe } 3 \end{cases}$$

(35)

Tabelle 11 zeigt die Werte der Parameter N_{j_a}, N_{j_b} und K_{j_b} für die beiden n-GR-Gruppen.

Tabelle 11: Parameter N_{j_a}, N_{j_b}, K_{j_b} für *n*-GR-Gruppe 2 und 3

Parameter	Gruppe 2	Gruppe 3
N_{j_a}	$n-3^i-2\cdot\sum_{k=1}^{j}3^{i-k}$	$3^{i-j}-3^{i-(i_{\min}-1)}+r$
N_{j_b}	$2\cdot 3^i-n$	$3^{i-(i_{\min}-1)}-r$
K_{j_b}	$2\cdot 3^{j-1}$	3^j

Jetzt werden die jeweiligen Parameter aus **Tabelle 11** und der XOR-Aufwand $\#\overset{iterativ}{XOR}_{\text{Teil }0}$ der Berechnung des Teils 0 für jede n-GR-Gruppe einzeln in Formel **(35**) eingesetzt, um den XOR-Aufwand $\#\overset{optimal}{XOR}_{c_{2n-2}^{n-GR}..c_n^{n-GR}}$ dieser Gruppen herzuleiten.

• *n*-GR aus Gruppe 2:

- Im Fall $i_{min}=1$:

$$\#\overset{optimal}{XOR}_{c_{2n-2}^{n-GR}..c_n^{n-GR}}=\#\overset{iterativ}{XOR}_{\text{Teil }0}+\#\overset{optimal}{XOR}_{\text{Teil }i_{\min}}=3^i-1+\#\overset{optimal}{XOR}_{\text{Teil }i_{\min}} \quad \textbf{(36)}$$

- Im Fall $i_{min}=2$:

$$\#\overset{optimal}{XOR}_{c_{2n-2}^{n-GR}..c_n^{n-GR}}=\#\overset{iterativ}{XOR}_{\text{Teil }0}+\#\overset{optimal}{XOR}_{\text{Teil }i_{\min}}+2\cdot N_{1\_a}+\left(2\cdot K_{1\_b}+2\right)\cdot N_{1\_b}-1=$$
$$=3^i-1+\#\overset{optimal}{XOR}_{\text{Teil }i_{\min}}+2\cdot\left(n-3^i-2\cdot 3^{i-1}\right)+\left(2\cdot 2+2\right)\cdot\left(2\cdot 3^i-n\right)-1=$$
$$=\#\overset{optimal}{XOR}_{\text{Teil }i_{\min}}-3^{i-1}+10\cdot 3^i-4\cdot n-2 \quad \textbf{(37)}$$

Für den Fall $i_{min}=2$ kann die Polynom-Länge *n* aus der Gruppe 2 folgendermaßen dargestellt werden:

$$n=3^i+2\cdot 3^i\underbrace{\sum_{j=1}^{i_{\min}-1}3^{-j}}_{\frac{1}{2}-\frac{1}{2\cdot 3^{i_{\min}-1}}}+r=2\cdot 3^i-3^{i-i_{\min}+1}+r=2\cdot 3^i-3^{i-1}+r \quad \textbf{(38)}$$

Jetzt kann Formel **(37)** unter der Berücksichtigung von **(38)** folgendermaßen vereinfacht werden:

$$\#\overset{optimal}{XOR}_{c_{2n-2}^{n-GR}..c_n^{n-GR}}=\#\overset{optimal}{XOR}_{\text{Teil }i_{\min}}-3^{i-1}+10\cdot 3^i-4\cdot\underbrace{n}_{=2\cdot 3^i-3^{i-1}+r}-2=$$
$$=\#\overset{optimal}{XOR}_{\text{Teil }i_{\min}}+3\cdot 3^i-4r-2$$

(39)

- Im Fall $i_{min}>2$

$$\#^{optimal}XOR_{c_{2n-2}^{n-GR}..c_n^{n-GR}} = \#^{iterativ}XOR_{\text{Teil }0} + \#^{optimal}XOR_{\text{Teil }i_{\min}} + 2\cdot N_{1\_a} + \left(2\cdot K_{1\_b}+3\right)\cdot N_{1\_b} -$$

$$-1 - N_{1\_b} + \sum_{j=2}^{i_{\min}-1}\left(2\cdot N_{j\_a} + \left(2\cdot K_{j\_b}+3\right)\cdot N_{j\_b}\right) =$$

$$= \#^{iterativ}XOR_{\text{Teil }0} + \#^{optimal}XOR_{\text{Teil }i_{\min}} - 1 - N_{1\_b} + \sum_{j=1}^{i_{\min}-1}\left(2\cdot N_{j\_a} + \left(2\cdot K_{j\_b}+3\right)\cdot N_{j\_b}\right) =$$

$$= \#^{optimal}XOR_{\text{Teil }i_{\min}} + \underbrace{3^i-1}_{\substack{iterativ\\ =\#XOR_{\text{Teil }0}}} - \left(\underbrace{2\cdot 3^i - n}_{=N_{1\_b}} + 1\right) +$$

$$+ \sum_{j=1}^{i_{\min}-1}\left(2\cdot\underbrace{\left(n - 3^i - 2\cdot\sum_{k=1}^{j}3^{i-k}\right)}_{=N_{j\_a}} + \left(3+4\cdot 3^{j-1}\right)\cdot\underbrace{\left(2\cdot 3^i - n\right)}_{=N_{j\_b}}\right) =$$

$$= \#^{optimal}XOR_{\text{Teil }i_{\min}} + 3^i - 1 -$$

$$-\left(2\cdot 3^i - n + 1\right) + \sum_{j=1}^{i_{\min}-1}\left(-4\cdot 3^i\cdot\underbrace{\sum_{k=1}^{j}3^{-k}}_{=\frac{1}{2}-\frac{1}{2\cdot 3^j}} + 4\cdot 3^i - n + 8\cdot 3^{i+j-1} - 4n\cdot 3^{j-1}\right) =$$

$$= \#^{optimal}XOR_{\text{Teil }i_{\min}} + 3^i - 1 -$$

$$-\left(2\cdot 3^i - n + 1\right) + \sum_{j=1}^{i_{\min}-1}\left(-4\cdot 3^i\cdot\left(\frac{1}{2}-\frac{1}{2\cdot 3^j}\right) + 4\cdot 3^i - n + 8\cdot 3^{i+j-1} - 4n\cdot 3^{j-1}\right) =$$

$$= \#^{optimal}XOR_{\text{Teil }i_{\min}} + 3^i - 1 - \left(2\cdot 3^i - n + 1\right) +$$

$$+ \sum_{j=1}^{i_{\min}-1}\left(-2\cdot 3^i\cdot\left(1-3^{-j}\right) + 4\cdot 3^i - n + 8\cdot 3^{i+j-1} - 4n\cdot 3^{j-1}\right) =$$

$$= \#^{optimal}XOR_{\text{Teil }i_{\min}} + 3^i - 1 - \left(2\cdot 3^i - n + 1\right) +$$

$$+ \sum_{j=1}^{i_{\min}-1}\left(2\cdot 3^{i-j} + \left(2\cdot 3^i - n\right) + \left(8\cdot 3^{i-1} - 4n\cdot 3^{-1}\right)\cdot 3^j\right) =$$

$$= \#^{optimal}XOR_{\text{Teil }i_{\min}} + 3^i - 1 - \left(2\cdot 3^i - n + 1\right) +$$

$$+ 2\cdot 3^i\cdot\underbrace{\sum_{j=1}^{i_{\min}-1}3^{-j}}_{=\frac{1}{2}-\frac{1}{2\cdot 3^{i_{\min}-1}}} + \left(2\cdot 3^i - n\right)\cdot\underbrace{\sum_{j=1}^{i_{\min}-1}1}_{=i_{\min}-1} + \left(8\cdot 3^{i-1} - \frac{4n}{3}\right)\cdot\underbrace{\sum_{j=1}^{i_{\min}-1}3^j}_{=3\cdot\frac{1-3^{i_{\min}-1}}{1-3}} =$$

$$
\begin{aligned}
&= \#^{optimal} XOR_{\text{Teil } i_{\min}} + 3^i - 1 - 1 + n \cdot \left(4 - 2 \cdot 3^{i_{\min}-1} - i_{\min}\right) + \\
&+ 3^i \cdot \left(-7 - 3^{1-i_{\min}} + 2 \cdot i_{\min} + 4 \cdot 3^{i_{\min}-1}\right) = \\
&= \#^{optimal} XOR_{\text{Teil } i_{\min}} + 3^i - 1 + \underbrace{n}_{=3^i + 2 \cdot \sum_{j=1}^{i_{\min}-1} 3^{i-j} + r = 2 \cdot 3^i - 3^{i-i_{\min}+1} + r} \cdot \left(4 - 2 \cdot 3^{i_{\min}-1} - i_{\min}\right) + \\
&3^i \cdot \left(-7 - 3^{1-i_{\min}} + 2 \cdot i_{\min} + 4 \cdot 3^{i_{\min}-1}\right) - 1 = \\
&= \#^{optimal} XOR_{\text{Teil } i_{\min}} + 4 \cdot 3^i + 3^{i-i_{\min}+1} \cdot \left(i_{\min} - 5\right) + 4r - 2r \cdot 3^{i_{\min}-1} - r \cdot i_{\min} - 2 = \\
&= \#^{optimal} XOR_{\text{Teil } i_{\min}} + 4 \cdot 3^i + 3^{i-i_{\min}+1} \cdot \left(i_{\min} - 5\right) - r\left(2 \cdot 3^{i_{\min}-1} + i_{\min} - 4\right) - 2
\end{aligned}
\qquad \textbf{(40)}
$$

• *n*-GR aus Gruppe 3:

- Im Fall $i_{min}=1$:

$$
\#^{optimal} XOR_{c_{2n-2}^{n-GR} .. c_n^{n-GR}} = \#^{iterativ} XOR_{\text{Teil } 0} + \#^{optimal} XOR_{\text{Teil } i_{\min}} = 2 \cdot 3^i - 1 + \#^{optimal} XOR_{\text{Teil } i_{\min}}
\qquad \textbf{(41)}
$$

- Im Fall $i_{min}>1$:

$$
\begin{aligned}
&\#^{optimal} XOR_{c_{2n-2}^{n-GR} .. c_n^{n-GR}} = \#^{iterativ} XOR_{\text{Teil } 0} + \#^{optimal} XOR_{\text{Teil } i_{\min}} + 2 \cdot N_{1\_a} + \left(2 \cdot K_{1\_b} + 3\right) \cdot N_{1\_b} - 1 + \\
&+ \sum_{j=2}^{i_{\min}-1} \left(2 \cdot N_{j\_a} + \left(2 \cdot K_{j\_b} + 3\right) \cdot N_{j\_b}\right) = \\
&= \#^{iterativ} XOR_{\text{Teil } 0} + \#^{optimal} XOR_{\text{Teil } i_{\min}} - 1 + \sum_{j=1}^{i_{\min}-1} \left(2 \cdot N_{j\_a} + \left(2 \cdot K_{j\_b} + 3\right) \cdot N_{j\_b}\right) = \\
&= \#^{optimal} XOR_{\text{Teil } i_{\min}} + \underbrace{2 \cdot 3^i - 1}_{=\#^{iterativ} XOR_{\text{Teil } 0}} - 1 + \\
&+ \sum_{j=1}^{i_{\min}-1} \left(2 \cdot \underbrace{\left(3^{i-j} - 3^{i-(i_{\min}-1)} + r\right)}_{=N_{j\_a}} + \left(3 + 2 \cdot \underbrace{3^j}_{=K_{j\_b}} \right) \cdot \underbrace{\left(3^{i-(i_{\min}-1)} - r\right)}_{=N_{j\_b}} \right) = \\
&= \#^{optimal} XOR_{\text{Teil } i_{\min}} + 2 \cdot 3^i - 2 + \sum_{j=1}^{i_{\min}-1} \left(2 \cdot 3^{i-j} + 3^{i-(i_{\min}-1)} - r + 2 \cdot 3^{i-(i_{\min}-1)} \cdot 3^j - 2r \cdot 3^j\right) = \\
&= \#^{optimal} XOR_{\text{Teil } i_{\min}} + 2 \cdot 3^i - 2 + 2 \cdot 3^i \cdot \underbrace{\sum_{j=1}^{i_{\min}-1} 3^{-j}}_{=\frac{1}{2} - \frac{1}{2 \cdot 3^{i_{\min}-1}}} + \left(3^{i-(i_{\min}-1)} - r\right) \cdot \underbrace{\sum_{j=1}^{i_{\min}-1} 1}_{=i_{\min}-1} +
\end{aligned}
$$

$$+2\cdot\left(3^{i-(i_{\min}-1)}-r\right)\cdot\underbrace{\sum_{j=1}^{i_{\min}-1}3^j}_{=3\cdot\frac{1-3^{i_{\min}-1}}{1-3}}=$$

$$=\#\overset{optimal}{XOR}_{\text{Teil }i_{\min}}+2\cdot3^i-2+3^i-2\cdot3^{i-(i_{\min}-1)}+\left(3^{i-(i_{\min}-1)}-r\right)\cdot i_{\min}+r+$$
$$+\left(3^{i-(i_{\min}-1)}-r\right)\cdot3\cdot\left(3^{i_{\min}-1}-1\right)=$$

$$=\#\overset{optimal}{XOR}_{\text{Teil }i_{\min}}+2\cdot3^i-2+3^i-2\cdot3^{i-i_{\min}+1}+\left(3^{i-i_{\min}+1}-r\right)\cdot i_{\min}+r+$$
$$+\left(3^{i-i_{\min}+1}-r\right)\cdot\left(3^{i_{\min}}-3\right)=$$
$$=\#\overset{optimal}{XOR}_{\text{Teil }i_{\min}}+2\cdot3^i-2+3^i-5\cdot3^{i-i_{\min}+1}+\left(3^{i-i_{\min}+1}-r\right)\cdot i_{\min}+3^{i+1}-$$
$$-r\cdot3^{i_{\min}}+4\cdot r=$$
$$=\#\overset{optimal}{XOR}_{\text{Teil }i_{\min}}+6\cdot3^i+3^{i-i_{\min}+1}\cdot\left(i_{\min}-5\right)-r\cdot\left(3^{i_{\min}}+i_{\min}-4\right)-2$$

(42)

Die Formeln **(36)**, **(39)**, **(40), (41)** und **(42)** können folgendermaßen zusammen gefasst werden:

$$\#\overset{optimal}{XOR}_{c_{2n-2}^{GR}\ldots c_n^{GR}}=\#\overset{optimal}{XOR}_{\text{Teil }i_{\min}}+$$
$$+\begin{cases}3^i-1, & n\text{-GR aus Gruppe 2, } i_{\min}=1\\ 2\cdot3^i-1, & n\text{-GR aus Gruppe 3, } i_{\min}=1\\ 4\cdot3^i+3^{i-i_{\min}+1}\cdot\left(i_{\min}-5\right)-r\left(2\cdot3^{i_{\min}-1}+i_{\min}-4\right)-2, & n\text{-GR aus Gruppe 2, } i_{\min}\geq2\\ 6\cdot3^i+3^{i-i_{\min}+1}\cdot\left(i_{\min}-5\right)-r\cdot\left(3^{i_{\min}}+i_{\min}-4\right)-2, & n\text{-GR aus Gruppe 3, } i_{\min}\geq2\end{cases}=$$

$$=\#\overset{optimal}{XOR}_{\text{Teil }i_{\min}}+\begin{cases}\begin{cases}3^i-1, & n\text{-GR aus Gruppe 2, } i_{\min}=1\\ 2\cdot3^i-1, & n\text{-GR aus Gruppe 3, } i_{\min}=1\end{cases}\\ 4\cdot3^i+3^{i-i_{\min}+1}\cdot\left(i_{\min}-5\right)-r\left(2\cdot3^{i_{\min}-1}+i_{\min}-4\right)-2+\\ \quad+\begin{cases}0, & n\text{-GR aus Gruppe 2, } i_{\min}\geq2\\ 2\cdot3^i-r\cdot3^{i_{\min}-1}, & n\text{-GR aus Gruppe 3, } i_{\min}\geq2\end{cases}\end{cases}$$

(43)

Anhang 3

Gatter-Komplexität der rekursiv angewendeten generalisierten Karatsuba-MM

Tabelle 12 zeigt die Gatter-Komplexität der folgenden MM:

- die in [18] veröffentlichten Werte für die rekursiv angewendete generalisierte Karatsuba-MM für Polynom-Längen bis 128 Bits
- die im Rahmen dieser Arbeit rekonstruierten Werte für die rekursiv angewendete generalisierte Karatsuba- MM für Polynom-Längen bis 600 Bits
- die in [18] veröffentlichten Werte für die rekursiv angewendete „simple“ Karatsuba-MM für Polynom-Längen bis 128 Bits
- die im Rahmen dieser Arbeit rekonstruierten Werte für die rekursiv angewendeten „simplen“ Karatsuba-MM für Polynom-Längen bis 600 Bits. Diese Werte stimmen für Polynom-Längen bis 128 Bits mit den Werten aus [18] überein. Sie wurden mittels **Algorithmus 6** unter der Verwendung der Formeln **(165)** und **(164)** (siehe Kapitel 5) ermittelt
- die im Rahmen dieser Arbeit ermittelten korrigierten Werte für die rekursiv angewendeten „simplen“ Karatsuba-MM für Polynom-Längen bis 600 Bits. Für Polynom-Längen bis 128 Bits unterscheiden sich diese Werte von den Werten aus [18]. Sie wurden mittels **Algorithmus 6** unter der Verwendung der Formeln **(163)** und **(164)** (siehe Kapitel 5) ermittelt
- Für die Auswahl der Werte in der letzten Spalte wurde für die rekursiv angewendeten generalisierten Karatsuba-MM und für die korrigierten Werte der „simplen“ Karatsuba-MM jeweils der Flächenbedarf in der IHP 0,13µ-Technologie berechnet. Es wurde jeweils die Gatter-Komplexität mit dem geringsten Flächenbedarf ausgewählt. Diese Werte wurden für die Evaluierung der Ergebnisse dieser Arbeit verwendet

Tabelle 12: Gatter-Komplexität der generalisierten und der „simplen“ Karatsuba-MM, rekursive Anwendung

n	Optimale nach [18] Faktorisierung von n	Rekursiv angewendete generalisierte Karatsuba-MM				Rekursiv angewendete “simple” Karatsuba-MM						Vergleichs-Daten: die jeweils Flächen-effizientesten Werte aus der rekursiven generalisierten und „simplen“ (korrigierte Werte) Karatsuba-MM	
		Daten aus [18]		Rekonstruierte Daten		Daten aus [18]		Rekonstruierte Daten					
								Gleiche Werte wie in [18]: nach den Formeln **(165)**, **(164)** aus Kapitel 5		Korrigierte Werte: nach den Formeln **(163)**, **(164)** aus Kapitel 5			
		#AND	#XOR	#AND	#XOR	#AND	#XOR	#AND	#XOR	#&	#XOR	#&	#XOR
2	=2	3	4	3	4	3	4	3	4	3	4	3	4
3	=3	6	13	6	13	6	13	6	13	6	14	6	13
4	=2*2	9	24	9	24	9	24	9	24	9	24	9	24
5	=5	15	46	15	46	15	46	15	46	14	46	14	46
6	=2*3	18	59	18	59	18	59	18	59	18	62	18	59
7	=7	28	99	28	99	24	85	24	85	23	84	23	84
8	=2*2*2	27	100	27	100	27	100	27	100	27	100	27	100
9	=3*3	36	139	36	139	39	148	39	148	36	146	36	139
10	=2*5	45	174	45	174	45	174	45	174	42	174	42	174
11	=11	66	265	66	265	51	204	51	204	49	208	49	208
12	=2*2*3	54	221	54	221	54	221	54	221	54	230	54	221
13	=13	91	378	91	378	66	277	66	277	63	276	63	276
14	=2*7	84	349	84	349	72	307	72	307	69	304	69	304
15	=3*5	90	385	90	385	78	341	78	341	76	338	76	338
16	=2*2*2*2	81	360	81	360	81	360	81	360	81	360	81	360
17	=17	153	664	153	664	105	460	105	460	98	454	98	454
18	=2*3*3	108	485	108	485	117	512	117	512	108	506	108	485
19	=19	190	837	190	837	129	568	129	568	119	564	119	564
20	=2*2*5	135	598	135	598	135	598	135	598	126	598	126	598
21	=3*7	168	751	168	751	147	662	147	662	139	668	139	668
22	=2*11	198	879	198	879	153	696	153	696	147	708	147	708
23	=23	276	1243	276	1243	159	734	159	734	156	754	156	754
24	=2*2*2*3	162	755	162	755	162	755	162	755	162	782	162	755
25	=5*5	225	1056	225	1056	186	871	186	871	179	876	179	876
26	=2*13	273	1234	273	1234	198	931	198	931	189	928	189	928
27	=3*3*3	216	1039	216	1039	210	995	210	995	200	986	200	986
28	=2*2*7	252	1155	252	1155	216	1029	216	1029	207	1020	207	1020
29	=29	435	2002	435	2002	228	1101	228	1101	220	1090	220	1090
30	=2*3*5	270	1271	270	1271	234	1139	234	1139	228	1130	228	1130
31	=31	496	2295	496	2295	240	1181	240	1181	237	1176	237	1176
32	=2*2*2*2*2	243	1204	243	1204	243	1204	243	1204	243	1204	243	1204
33	=3*11	396	1843	396	1843	291	1408	291	1408	276	1394	276	1394
34	=2*17	459	2124	459	2124	315	1512	315	1512	294	1494	294	1494
35	=5*7	420	2011	420	2011	339	1620	339	1620	313	1600	313	1600
36	=2*2*3*3	324	1595	324	1595	351	1676	351	1676	324	1658	324	1595
37	=37	703	3294	703	3294	375	1792	375	1792	345	1776	345	1776
38	=2*19	570	2659	570	2659	387	1852	387	1852	357	1840	357	1840
39	=3*13	546	2569	546	2569	399	1916	399	1916	370	1910	370	1910
40	=2*2*2*5	405	1950	405	1950	405	1950	405	1950	378	1950	378	1950
41	=41	861	4060	861	4060	429	2082	429	2082	403	2092	403	2092
42	=2*3*7	504	2417	504	2417	441	2150	441	2150	417	2168	417	2168
43	=43	946	4473	946	4473	453	2222	453	2222	432	2250	432	2250
44	=2*2*11	594	2809	594	2809	459	2260	459	2260	441	2296	441	2296
45	=3*3*5	540	2659	540	2659	471	2340	471	2340	458	2390	458	2390
46	=2*23	828	3909	828	3909	477	2382	477	2382	468	2442	468	2442
47	=47	1128	5359	1128	5359	483	2428	483	2428	479	2500	479	2500
48	=2*2*2*2*3	486	2453	486	2453	486	2453	486	2453	486	2534	486	2453
49	=7*7	784	3879	784	3879	534	2689	534	2689	519	2724	519	2724
50	=2*5*5	675	3364	675	3364	558	2809	558	2809	537	2824	537	2824
51	=3*17	918	4381	918	4381	582	2933	582	2933	556	2930	556	2930
52	=2*2*13	819	3906	819	3906	594	2997	594	2997	567	2988	567	2988
53	=53	1431	6838	1431	6838	618	3129	618	3129	588	3106	588	3106
54	=2*3*3*3	648	3329	648	3329	630	3197	630	3197	600	3170	600	3170
55	=5*11	990	4821	990	4821	642	3269	642	3269	613	3240	613	3240
56	=2*2*2*7	756	3685	756	3685	648	3307	648	3307	621	3280	621	3280
57	=3*19	1140	5467	1140	5467	672	3455	672	3455	646	3422	646	3422
58	=2*29	1305	6234	1305	6234	684	3531	684	3531	660	3498	660	3498
59	=59	1770	8497	1770	8497	696	3611	696	3611	675	3580	675	3580
60	=2*2*3*5	810	4049	810	4049	702	3653	702	3653	684	3626	684	3626
61	=61	1891	9090	1891	9090	714	3741	714	3741	701	3720	701	3720
62	=2*31	1488	7129	1488	7129	720	3787	720	3787	711	3772	711	3772
63	=3*3*7	1008	4999	1008	4999	726	3837	726	3837	722	3830	722	3830
64	=2*2*2*2*2*2	729	3864	729	3864	729	3864	729	3864	729	3864	729	3864

65	=5*13	1365	6676	1365	6676	825	4276	825	4276	794	4246	794	4246
66	=2*3*11	1188	5789	1188	5789	873	4484	873	4484	828	4442	828	4442
67	=67	2278	10989	2278	10989	921	4696	921	4696	863	4644	863	4644
68	=2*2*17	1377	6640	1377	6640	945	4804	945	4804	882	4750	882	4750
69	=3*23	1656	7999	1656	7999	993	5024	993	5024	919	4964	919	4964
70	=2*5*7	1260	6309	1260	6309	1017	5136	1017	5136	939	5076	939	5076
71	=71	2556	12355	2556	12355	1041	5252	1041	5252	960	5194	960	5194
72	=2*2*2*3*3	972	5069	972	5069	1053	5312	1053	5312	972	5258	972	5069
73	=73	2701	13068	2701	13068	1101	5548	1101	5548	1013	5496	1013	5496
74	=2*37	2109	10174	2109	10174	1125	5668	1125	5668	1035	5620	1035	5620
75	=3*5*5	1350	6925	1350	6925	1149	5792	1149	5792	1058	5750	1058	5750
76	=2*2*19	1710	8277	1710	8277	1161	5856	1161	5856	1071	5820	1071	5820
77	=7*11	1848	9199	1848	9199	1185	5988	1185	5988	1096	5962	1096	5962
78	=2*3*13	1638	8015	1638	8015	1197	6056	1197	6056	1110	6038	1110	6038
79	=79	3160	15327	3160	15327	1209	6128	1209	6128	1125	6120	1125	6120
80	=2*2*2*2*5	1215	6166	1215	6166	1215	6166	1215	6166	1134	6166	1134	6166
81	=3*3*3*3	1296	6871	1296	6871	1263	6434	1263	6434	1183	6452	1183	6452
82	=2*41	2583	12504	2583	12504	1287	6570	1287	6570	1209	6600	1209	6600
83	=83	3486	16933	3486	16933	1311	6710	1311	6710	1236	6754	1236	6754
84	=2*2*3*7	1512	7583	1512	7583	1323	6782	1323	6782	1251	6836	1251	6836
85	=5*17	2295	11286	2295	11286	1347	6930	1347	6930	1280	7002	1280	7002
86	=2*43	2838	13759	2838	13759	1359	7006	1359	7006	1296	7090	1296	7090
87	=3*29	2610	12697	2610	12697	1371	7086	1371	7086	1313	7184	1313	7184
88	=2*2*2*11	1782	8775	1782	8775	1377	7128	1377	7128	1323	7236	1323	7236
89	=89	4005	19492	4005	19492	1401	7292	1401	7292	1356	7426	1356	7426
90	=2*3*3*5	1620	8333	1620	8333	1413	7376	1413	7376	1374	7526	1374	7526
91	=7*13	2548	12699	2548	12699	1425	7464	1425	7464	1393	7632	1393	7632
92	=2*2*23	2484	12091	2484	12091	1431	7510	1431	7510	1404	7690	1404	7690
93	=3*31	2976	14503	2976	14503	1443	7606	1443	7606	1425	7808	1425	7808
94	=2*47	3384	16449	3384	16449	1449	7656	1449	7656	1437	7872	1437	7872
95	=5*19	2850	14041	2850	14041	1455	7710	1455	7710	1450	7942	1450	7942
96	=2*2*2*2*2*3	1458	7739	1458	7739	1458	7739	1458	7739	1458	7982	1458	7739
97	=97	4753	23184	4753	23184	1554	8215	1554	8215	1523	8364	1523	8364
98	=2*7*7	2352	12025	2352	12025	1602	8455	1602	8455	1557	8560	1557	8560
99	=3*3*11	2376	11839	2376	11839	1650	8699	1650	8699	1592	8762	1592	8762
100	=2*2*5*5	2025	10488	2025	10488	1674	8823	1674	8823	1611	8868	1611	8868
101	=101	5151	25150	5151	25150	1722	9075	1722	9075	1648	9082	1648	9082
102	=2*3*17	2754	13547	2754	13547	1746	9203	1746	9203	1668	9194	1668	9194
103	=103	5356	26163	5356	26163	1770	9335	1770	9335	1689	9312	1689	9312
104	=2*2*2*13	2457	12130	2457	12130	1782	9403	1782	9403	1701	9376	1701	9376
105	=3*5*7	2520	12895	2520	12895	1830	9671	1830	9671	1742	9614	1742	9614
106	=2*53	4293	20934	4293	20934	1854	9807	1854	9807	1764	9738	1764	9738
107	=107	5778	28249	5778	28249	1878	9947	1878	9947	1787	9868	1787	9868
108	=2*2*3*3*3	1944	10415	1944	10415	1890	10019	1890	10019	1800	9938	1800	9938
109	=109	5995	29322	5995	29322	1914	10167	1914	10167	1825	10080	1825	10080
110	=2*5*11	2970	14899	2970	14899	1926	10243	1926	10243	1839	10156	1839	10156
111	=3*37	4218	20641	4218	20641	1938	10323	1938	10323	1854	10238	1854	10238
112	=2*2*2*2*7	2268	11499	2268	11499	1944	10365	1944	10365	1863	10284	1863	10284
113	=113	6441	31528	6441	31528	1992	10665	1992	10665	1912	10570	1912	10570
114	=2*3*19	3420	16853	3420	16853	2016	10817	2016	10817	1938	10718	1938	10718
115	=5*23	4140	20451	4140	20451	2040	10973	2040	10973	1965	10872	1965	10872
116	=2*2*29	3915	19162	3915	19162	2052	11053	2052	11053	1980	10954	1980	10954
117	=3*3*13	3276	16339	3276	16339	2076	11217	2076	11217	2009	11120	2009	11120
118	=2*59	5310	25959	5310	25959	2088	11301	2088	11301	2025	11208	2025	11208
119	=7*17	4284	21379	4284	21379	2100	11389	2100	11389	2042	11302	2042	11302
120	=2*2*2*3*5	2430	12623	2430	12623	2106	11435	2106	11435	2052	11354	2052	11354
121	=11*11	4356	22155	4356	22155	2130	11615	2130	11615	2085	11544	2085	11544
122	=2*61	5673	27754	5673	27754	2142	11707	2142	11707	2103	11644	2103	11644
123	=3*41	5166	25333	5166	25333	2154	11803	2154	11803	2122	11750	2122	11750
124	=2*2*31	4464	21879	4464	21879	2160	11853	2160	11853	2133	11808	2133	11808
125	=5*5*5	3375	17806	3375	17806	2172	11957	2172	11957	2154	11926	2154	11926
126	=2*3*3*7	3024	15497	3024	15497	2178	12011	2178	12011	2166	11990	2166	11990
127	=127	8128	39879	8128	39879	2184	12069	2184	12069	2179	12060	2179	12060
128	=2*2*2*2*2*2*2	2187	12100	2187	12100	2187	12100	2187	12100	2187	12100	2187	12100
129	=3*43			5676	27859			2379	12928	2316	12866	2316	12866
130	=2*5*13			4095	20544			2475	13344	2382	13254	2382	13254
131	=131			8646	42445			2571	13764	2449	13648	2449	13648
132	=2*2*3*11			3564	17891			2619	13976	2484	13850	2484	13850
133	=7*19			5320	26559			2715	14404	2553	14256	2553	14256
134	=2*67			6834	33499			2763	14620	2589	14464	2589	14464
135	=3*3*3*5			3240	17023			2811	14840	2626	14678	2626	14678
136	=2*2*2*17			4131	20460			2835	14952	2646	14790	2646	14790
137	=137			9453	46444			2931	15396	2719	15220	2719	15220
138	=2*3*23			4968	24545			2979	15620	2757	15440	2757	15440
139	=139			9730	47817			3027	15848	2796	15666	2796	15666
140	=2*2*5*7			3780	19483			3051	15964	2817	15784	2817	15784
141	=3*47			6768	33271			3099	16200	2858	16022	2858	16022
142	=2*71			7668	37629			3123	16320	2880	16146	2880	16146

143	=11*13			6006	30493			3147	16444	2903	16276	2903	16276
144	=2*2*2*2*3*3			2916	15779			3159	16508	2916	16346	2916	15779
145	=5*29			6525	32316			3255	16984	2997	16824	2997	16824
146	=2*73			8103	39784			3303	17224	3039	17068	3039	17068
147	=3*7*7			4704	24439			3351	17468	3082	17318	3082	17318
148	=2*2*37			6327	31110			3375	17592	3105	17448	3105	17448
149	=149			11175	54982			3423	17844	3150	17710	3150	17710
150	=2*3*5*5			4050	21371			3447	17972	3174	17846	3174	17846
151	=151			11476	56475			3471	18104	3199	17988	3199	17988
152	=2*2*2*19			5130	25435			3483	18172	3213	18064	3213	18064
153	=3*3*17			5508	27499			3531	18440	3262	18350	3262	18350
154	=2*7*11			5544	28209			3555	18576	3288	18498	3288	18498
155	=5*31			7440	36871			3579	18716	3315	18652	3315	18652
156	=2*2*3*13			4914	24665			3591	18788	3330	18734	3330	18734
157	=157			12403	61074			3615	18936	3359	18900	3359	18900
158	=2*79			9480	46609			3627	19012	3375	18988	3375	18988
159	=3*53			8586	42289			3639	19092	3392	19082	3392	19082
160	=2*2*2*2*2*5			3645	19134			3645	19134	3402	19134	3402	19134
161	=7*23			7728	38599			3741	19674	3499	19708	3499	19708
162	=2*3*3*3*3			3888	21257			3789	19946	3549	20000	3549	20000
163	**=163**			**13366**	**65853**			3837	20222	3600	20298	**3600**	**20298**
164	**=2*2*41**			7749	38164			3861	20362	3627	20452	3627	20452
165	**=3*5*11**			5940	30235			3909	20646	3680	20762	3680	20762
166	=2*83			10458	51459			3933	20790	3708	20922	3708	20922
167	=167			14028	69139			3957	20938	3737	21088	3737	21088
168	=2*2*2*3*7			4536	23417			3969	21014	3753	21176	3753	21176
169	=13*13			8281	42264			4017	21314	3810	21510	3810	21510
170	=2*5*17			6885	34534			4041	21466	3840	21682	3840	21682
171	=3*3*19			6840	34159			4065	21622	3871	21860	3871	21860
172	=2*2*43			8514	41961			4077	21702	3888	21954	3888	21954
173	=173			15051	74218			4101	21866	3921	22144	3921	22144
174	=2*3*29			7830	38783			4113	21950	3939	22244	3939	22244
175	=5*5*7			6300	32931			4125	22038	3958	22350	3958	22350
176	=2*2*2*2*11			5346	27025			4131	22084	3969	22408	3969	22408
177	=3*59			10620	52387			4179	22416	4034	22790	4034	22790
178	=2*89			12015	59184			4203	22584	4068	22986	4068	22986
179	=179			16110	79477			4227	22756	4103	23188	4103	23188
180	=2*2*3*3*5			4860	25715			4239	22844	4122	23294	4122	23294
181	=181			16471	81270			4263	23024	4159	23508	4159	23508
182	=2*7*13			7644	38821			4275	23116	4179	23620	4179	23620
183	=3*61			11346	55993			4287	23212	4200	23738	4200	23738
184	=2*2*2*23			7452	37005			4293	23262	4212	23802	4212	23802
185	=5*37			10545	52336			4317	23458	4253	24040	4253	24040
186	=2*3*31			8928	44249			4329	23558	4275	24164	4275	24164
187	=11*17			10098	51129			4341	23662	4298	24294	4298	24294
188	=2*2*47			10152	50095			4347	23716	4311	24364	4311	24364
189	=3*3*3*7			6048	31495			4359	23828	4336	24506	4336	24506
190	=2*5*19			8550	42879			4365	23886	4350	24582	4350	24582
191	=191			18336	90535			4371	23948	4365	24664	4365	24664
192	=2*2*2*2*2*2*3			4374	23981			4374	23981	4374	24710	4374	23981
193	=193			18721	92448			4566	24937	4503	25476	4503	25476
194	=2*97			14259	70324			4662	25417	4569	25864	4569	25864
195	=3*5*13			8190	41605			4758	25901	4636	26258	4636	26258
196	=2*2*7*7			7056	36855			4806	26145	4671	26460	4671	26460
197	=197			19503	96334			4902	26637	4740	26866	4740	26866
198	=2*3*3*11			7128	36305			4950	26885	4776	27074	4776	27074
199	=199			19900	98307			4998	27137	4813	27288	4813	27288
200	=2*2*2*5*5			6075	32260			5022	27265	4833	27400	4833	27400
201	=3*67			13668	67531			5118	27773	4906	27830	4906	27830
202	=2*101			15453	76254			5166	28029	4944	28050	4944	28050
203	=7*29			12180	60859			5214	28289	4983	28276	4983	28276
204	=2*2*3*17			8262	41453			5238	28421	5004	28394	5004	28394
205	=5*41			12915	64146			5286	28689	5045	28632	5045	28632
206	=2*103			16068	79309			5310	28825	5067	28756	5067	28756
207	=3*3*23			9936	49639			5334	28965	5090	28886	5090	28886
208	=2*2*2*2*13			7371	37218			5346	29037	5103	28956	5103	28956
209	=11*19			12540	63427			5442	29577	5184	29434	5184	29434
210	=2*3*5*7			7560	39521			5490	29849	5226	29678	5226	29678
211	=211			22366	110565			5538	30125	5269	29928	5269	29928
212	=2*2*53			12879	63646			5562	30265	5292	30058	5292	30058
213	=3*71			15336	75823			5610	30549	5337	30320	5337	30320
214	=2*107			17334	85599			5634	30693	5361	30456	5361	30456
215	=5*43			14190	70501			5658	30841	5386	30598	5386	30598
216	=2*2*2*3*3*3			5832	32105			5670	30917	5400	30674	5400	30674
217	=7*31			13888	69399			5718	31217	5449	30960	5449	30960
218	=2*109			17985	88834			5742	31369	5475	31108	5475	31108
219	=3*73			16206	80149			5766	31525	5502	31262	5502	31262
220	=2*2*5*11			8910	45573			5778	31605	5517	31344	5517	31344

221	=13*17			13923	70786			5802	31769	5546	31510	5546	31510
222	=2*3*37			12654	62807			5814	31853	5562	31598	5562	31598
223	=223			24976	123543			5826	31941	5579	31692	5579	31692
224	=2*2*2*2*2*7			6804	35389			5832	31987	5589	31744	5589	31744
225	=3*3*5*5			8100	43339			5928	32591	5686	32318	5686	32318
226	=2*113			19323	95484			5976	32895	5736	32610	5736	32610
227	=227			25878	128029			6024	33203	5787	32908	5787	32908
228	=2*2*3*19			10260	51467			6048	33359	5814	33062	5814	33062
229	=229			26335	130302			6096	33675	5867	33372	5867	33372
230	=2*5*23			12420	62269			6120	33835	5895	33532	5895	33532
231	=3*7*11			11088	57031			6144	33999	5924	33698	5924	33698
232	=2*2*2*29			11745	58410			6156	34083	5940	33786	5940	33786
233	=233			27261	134908			6204	34415	5997	34120	**5997**	**34120**
234	=2*3*3*13			9828	49949			6228	34583	6027	34292	6027	34292
235	=5*47			16920	84111			6252	34755	6058	34470	6058	34470
236	=2*2*59			15930	78817			6264	34843	6075	34564	6075	34564
237	=3*79			18960	93847			6288	35023	6108	34754	6108	34754
238	=2*7*17			12852	65085			6300	35115	6126	34854	6126	34854
239	=239			28680	141967			6312	35211	6145	34960	6145	34960
240	=2*2*2*2*3*5			7290	38825			6318	35261	6156	35018	6156	35018
241	=241			29161	144360			6366	35625	6221	35400	6221	35400
242	=2*11*11			13068	67429			6390	35809	6255	35596	6255	35596
243	=3*3*3*3*3			7776	43159			6414	35997	6290	35798	6290	35798
244	=2*2*61			17019	84234			6426	36093	6309	35904	6309	35904
245	=5*7*7			11760	62071			6450	36289	6346	36118	6346	36118
246	=2*3*41			15498	76979			6462	36389	6366	36230	6366	36230
247	=13*19			17290	87777			6474	36493	6387	36348	6387	36348
248	=2*2*2*31			13392	66625			6480	36547	6399	36412	6399	36412
249	=3*83			20916	103579			6504	36759	6440	36650	6440	36650
250	=2*5*5*5			10125	54414			6516	36867	6462	36774	6462	36774
251	=251			31626	156625			6528	36979	6485	36904	6485	36904
252	=2*2*3*3*7			9072	47495			6534	37037	6498	36974	6498	36974
253	=11*23			18216	91983			6546	37157	6523	37116	6523	37116
254	=2*127			24384	120649			6552	37219	6537	37192	6537	37192
255	=3*5*17			13770	69745			6558	37285	6552	37274	6552	37274
256	=2*2*2*2*2*2*2*2			6561	37320			6561	37320	6561	37320	6561	37320
257	=257			33153	164224			6945	38980	6818	38854	6818	38854
258	=2*3*43			17028	84605			7137	39812	6948	39626	6948	39626
259	=7*37			19684	98379			7329	40648	7079	40404	7079	40404
260	=2*2*5*13			12285	62668			7425	41068	7146	40798	7146	40798
261	=3*3*29			15660	78259			7617	41912	7279	41588	7279	41588
262	=2*131			25938	128379			7713	42336	7347	41988	7347	41988
263	=263			34716	172003			7809	42764	7416	42394	7416	42394
264	=2*2*2*3*11			10692	54725			7857	42980	7452	42602	7452	42602
265	=5*53			21465	106776			8049	43840	7589	43416	7589	43416
266	=2*7*19			15960	80737			8145	44272	7659	43828	7659	43828
267	=3*89			24030	119077			8241	44708	7730	44246	7730	44246
268	=2*2*67			20502	101565			8289	44928	7767	44460	7767	44460
269	=269			36315	179962			8385	45372	7840	44890	7840	44890
270	=2*3*3*3*5			9720	52145			8433	45596	7878	45110	7878	45110
271	=271			36856	182655			8481	45824	7917	45336	7917	45336
272	=2*2*2*2*17			12393	62464			8505	45940	7938	45454	7938	45454
273	=3*7*13			15288	78367			8697	46832	8083	46316	8083	46316
274	=2*137			28359	140424			8793	47280	8157	46752	8157	46752
275	=5*5*11			14850	76681			8889	47732	8232	47194	8232	47194
276	=2*2*3*23			14904	74735			8937	47960	8271	47420	8271	47420
277	=277			38503	190854			9033	48420	8348	47874	8348	47874
278	=2*139			29190	144559			9081	48652	8388	48106	8388	48106
279	=3*3*31			17856	89239			9129	48888	8429	48344	8429	48344
280	=2*2*2*5*7			11340	59565			9153	49008	8451	48468	8451	48468
281	=281			39621	196420			9249	49484	8532	48946	8532	48946
282	=2*3*47			20304	100937			9297	49724	8574	49190	8574	49190
283	=283			40186	199233			9345	49968	8617	49440	**8617**	**49440**
284	=2*2*71			23004	114019			9369	50092	8640	49570	8640	49570
285	=3*5*19			17100	86515			9417	50344	8685	49832	8685	49832
286	=2*11*13			18018	92619			9441	50472	8709	49968	8709	49968
287	=7*41			24108	120499			9465	50604	8734	50110	8734	50110
288	=2*2*2*2*2*3*3			8748	48485			9477	50672	8748	50186	8748	48485
289	=17*17			23409	119664			9669	51628	8909	51144	8909	51144
290	=2*5*29			19575	98104			9765	52108	8991	51628	8991	51628
291	=3*97			28518	141421			9861	52592	9074	52118	9074	52118
292	=2*2*73			24309	120516			9909	52836	9117	52368	9117	52368
293	=293			43071	213598			10005	53328	9202	52870	9202	52870
294	=2*3*7*7			14112	74489			10053	53576	9246	53126	9246	53126
295	=5*59			26550	132141			10101	53828	9291	53388	9291	53388
296	=2*2*2*37			18981	94510			10125	53956	9315	53524	9315	53524
297	=3*3*3*11			14256	73399			10221	54464	9404	54050	9404	54050
298	=2*149			33525	166134			10269	54720	9450	54318	9450	54318

299	=13*23			25116	127219			10317	54980	9497	54592	9497	54592
300	=2*2*3*5*5			12150	65309			10341	55112	9522	54734	9522	54734
301	=7*43			26488	132399			10389	55380	9571	55020	9571	55020
302	=2*151			34428	170629			10413	55516	9597	55168	9597	55168
303	=3*101			30906	153313			10437	55656	9624	55322	9624	55322
304	=2*2*2*2*19			15390	77517			10449	55728	9639	55404	9639	55404
305	=5*61			28365	141196			10545	56268	9736	55978	9736	55978
306	=2*3*3*17			16524	83717			10593	56540	9786	56270	9786	56270
307	=307			47278	234549			10641	56816	9837	56568	9837	56568
308	=2*2*7*11			16632	85855			10665	56956	9864	56722	9864	56722
309	=3*103			32136	159439			10713	57240	9917	57032	9917	57032
310	=2*5*31			22320	111849			10737	57384	9945	57192	9945	57192
311	=311			48516	240715			10761	57532	9974	57358	9974	57358
312	=2*2*2*3*13			14742	75239			10773	57608	9990	57446	9990	57446
313	=313			49141	243828			10821	57908	10047	57780	10047	57780
314	=2*157			37209	184474			10845	58060	10077	57952	10077	57952
315	=3*3*5*7			15120	79879			10869	58216	10108	58130	10108	58130
316	=2*2*79			28440	141087			10881	58296	10125	58224	10125	58224
317	=317			50403	250114			10905	58460	10158	58414	10158	58414
318	=2*3*53			25758	128135			10917	58544	10176	58514	10176	58514
319	=11*29			28710	144717			10929	58632	10195	58620	10195	58620
320	=2*2*2*2*2*2*5			10935	58678			10935	58678	10206	58678	10206	58678
321	=3*107			34668	172051			11127	59762	10399	59828	10399	59828
322	=2*7*23			23184	117081			11223	60306	10497	60408	10497	60408
323	=17*19			29070	148309			11319	60854	10596	60994	10596	60994
324	=2*2*3*3*3*3			11664	65063			11367	61130	10647	61292	10647	61292
325	=5*5*13			20475	105306			11463	61686	10748	61890	10748	61890
326	=2*163			40098	198859			11511	61966	10800	62194	10800	62194
327	=3*109			35970	178537			11559	62250	10853	62504	10853	62504
328	=2*2*2*41			23247	115800			11583	62394	10881	62664	10881	62664
329	=7*47			31584	157879			11679	62966	10986	63286	10986	63286
330	=2*3*5*11			17820	92021			11727	63254	11040	63602	11040	63602
331	=331			54946	272745			11775	63546	11095	63924	11095	63924
332	=2*2*83			31374	155701			11799	63694	11124	64090	11124	64090
333	=3*3*37			25308	126499			11847	63994	11181	64424	11181	64424
334	=2*167			42084	208749			11871	64146	11211	64596	11211	64596
335	=5*67			34170	170161			11895	64302	11242	64774	11242	64774
336	=2*2*2*2*3*7			13608	71591			11907	64382	11259	64868	11259	64868
337	=337			56953	282744			12003	64986	11372	65538	11372	65538
338	=2*13*13			24843	128140			12051	65290	11430	65878	11430	65878
339	=3*113			38646	191869			12099	65598	11489	66224	11489	66224
340	=2*2*5*17			20655	104958			12123	65754	11520	66402	11520	66402
341	=11*31			32736	164935			12171	66070	11581	66760	11581	66760
342	=2*3*3*19			20520	103841			12195	66230	11613	66944	11613	66944
343	=7*7*7			21952	116775			12219	66394	11646	67134	11646	67134
344	=2*2*2*43			25542	127255			12231	66478	11664	67234	11664	67234
345	=3*5*23			24840	125455			12279	66810	11729	67616	11729	67616
346	=2*173			45153	224034			12303	66978	11763	67812	11763	67812
347	=347			60378	299809			12327	67150	11798	68014	11798	68014
348	=2*2*3*29			23490	117737			12339	67238	11817	68120	11817	68120
349	=349			61075	303282			12363	67418	11854	68334	11854	68334
350	=2*5*5*7			18900	100189			12375	67510	11874	68446	11874	68446
351	=3*3*3*13			19656	100831			12387	67606	11895	68564	11895	68564
352	=2*2*2*2*2*11			16038	82479			12393	67656	11907	68628	11907	68628
353	=353			62481	310288			12489	68324	12036	69394	12036	69394
354	=2*3*59			31860	158573			12537	68660	12102	69782	12102	69782
355	=5*71			38340	190971			12585	69000	12169	70176	12169	70176
356	=2*2*89			36045	178972			12609	69172	12204	70378	12204	70378
357	=3*7*17			25704	131119			12657	69520	12273	70784	12273	70784
358	=2*179			48330	239859			12681	69696	12309	70992	12309	70992
359	=359			64620	320947			12705	69876	12346	71206	12346	71206
360	=2*2*2*3*3*5			14580	78581			12717	69968	12366	71318	12366	71318
361	=19*19			36100	184491			12765	70332	12439	71748	12439	71748
362	=2*181			49413	245254			12789	70516	12477	71968	12477	71968
363	=3*11*11			26136	135823			12813	70704	12516	72194	12516	72194
364	=2*2*7*13			22932	117915			12825	70800	12537	72312	12537	72312
365	=5*73			40515	201826			12849	70996	12578	72550	12578	72550
366	=2*3*61			34038	169439			12861	71096	12600	72674	12600	72674
367	=367			67528	335439			12873	71200	12623	72804	12623	72804
368	=2*2*2*2*23			22356	112483			12879	71254	12636	72874	12636	72874
369	=3*3*41			30996	154939			12927	71650	12717	73352	12717	73352
370	=2*5*37			31635	158484			12951	71850	12759	73596	12759	73596
371	=7*53			40068	200299			12975	72054	12802	73846	12802	73846
372	=2*2*3*31			26784	134231			12987	72158	12825	73976	12825	73976
373	=373			69751	346518			13011	72370	12870	74238	12870	74238
374	=2*11*17			30294	154879			13023	72478	12894	74374	12894	74374
375	=3*5*5*5			20250	109825			13035	72590	12919	74516	12919	74516
376	=2*2*2*47			30456	151785			13041	72648	12933	74592	12933	74592

377	=13*29			39585	200032			13065	72876	12982	74878	12982	74878
378	=2*3*3*3*7			18144	95993			13077	72992	13008	75026	13008	75026
379	=379			72010	357777			13089	73112	13035	75180	13035	75180
380	=2*2*5*19			25650	130153			13095	73174	13050	75262	13050	75262
381	=3*127			48768	242311			13107	73302	13079	75428	13079	75428
382	=2*191			55008	273129			13113	73368	13095	75516	13095	75516
383	=383			73536	365383			13119	73438	13112	75610	13112	75610
384	=2*2*2*2*2*2*2*3			13122	73475			13122	73475	13122	75662	13122	73475
385	=5*7*11			27720	144111			13506	75391	13379	77196	13379	77196
386	=2*193			56163	278884			13698	76351	13509	77968	13509	77968
387	=3*3*43			34056	170239			13890	77315	13640	78746	13640	78746
388	=2*2*97			42777	212520			13986	77799	13707	79140	13707	79140
389	=389			75855	376942			14178	78771	13840	79930	13840	79930
390	=2*3*5*13			24570	126371			14274	79259	13908	80330	13908	80330
391	=17*23			42228	214779			14370	79751	13977	80736	13977	80736
392	=2*2*2*7*7			21168	112129			14418	79999	14013	80944	14013	80944
393	=3*131			51876	257803			14610	80987	14150	81758	14150	81758
394	=2*197			58509	290574			14706	81483	14220	82170	14220	82170
395	=5*79			47400	236191			14802	81983	14291	82588	14291	82588
396	=2*2*3*3*11			21384	110495			14850	82235	14328	82802	14328	82802
397	=397			79003	392634			14946	82743	14401	83232	14401	83232
398	=2*199			59700	296509			14994	82999	14439	83452	14439	83452
399	=3*7*19			31920	162535			15042	83259	14478	83678	14478	83678
400	=2*2*2*2*5*5			18225	98376			15066	83391	14499	83796	14499	83796
401	=401			80601	400600			15258	84411	14644	84658	14644	84658
402	=2*3*67			41004	204197			15354	84923	14718	85094	14718	85094
403	=13*31			45136	227943			15450	85439	14793	85536	14793	85536
404	=2*2*101			46359	230374			15498	85699	14832	85762	14832	85762
405	=3*3*3*3*5			19440	105367			15594	86223	14909	86216	14909	86216
406	=2*7*29			36540	184197			15642	86487	14949	86448	14949	86448
407	=11*37			46398	233509			15690	86755	14990	86686	14990	86686
408	=2*2*2*3*17			24786	125987			15714	86891	15012	86810	15012	86810
409	=409			83845	416772			15810	87431	15093	87288	**15093**	**87288**
410	=2*5*41			38745	194074			15858	87703	15135	87532	15135	87532
411	=3*137			56718	281941			15906	87979	15178	87782	15178	87782
412	=2*2*103			48204	239571			15930	88119	15201	87912	15201	87912
413	=7*59			49560	247759			15978	88403	15246	88174	15246	88174
414	=2*3*3*23			29808	150569			16002	88547	15270	88310	15270	88310
415	=5*83			52290	260601			16026	88695	15295	88452	15295	88452
416	=2*2*2*2*2*13			22113	113314			16038	88771	15309	88528	15309	88528
417	=3*139			58380	290227			16230	89855	15470	89486	15470	89486
418	=2*11*19			37620	191949			16326	90399	15552	89970	15552	89970
419	=419			87990	437437			16422	90947	15635	90460	15635	90460
420	=2*2*3*5*7			22680	120239			16470	91223	15678	90710	15678	90710
421	=421			88831	441630			16566	91779	15763	91212	15763	91212
422	=2*211			67098	333379			16614	92059	15807	91468	15807	91468
423	=3*3*47			40608	202999			16662	92343	15852	91730	15852	91730
424	=2*2*2*53			38637	192630			16686	92487	15876	91866	15876	91866
425	=5*5*17			34425	176056			16782	93059	15965	92392	15965	92392
426	=2*3*71			46008	229169			16830	93347	16011	92660	16011	92660
427	=7*61			52948	264699			16878	93639	16058	92934	16058	92934
428	=2*2*107			52002	258505			16902	93787	16083	93076	16083	93076
429	=3*11*13			36036	186379			16950	94087	16132	93362	16132	93362
430	=2*5*43			42570	213219			16974	94239	16158	93510	16158	93510
431	=431			93096	462895			16998	94395	16185	93664	16185	93664
432	=2*2*2*2*3*3*3			17496	98039			17010	94475	16200	93746	16200	93746
433	=433			93961	467208			17106	95079	16297	94320	16297	94320
434	=2*7*31			41664	209929			17154	95383	16347	94612	16347	94612
435	=3*5*29			39150	197365			17202	95691	16398	94910	16398	94910
436	=2*2*109			53955	268242			17226	95847	16425	95064	16425	95064
437	=19*23			52440	267103			17274	96163	16478	95374	16478	95374
438	=2*3*73			48618	242195			17298	96323	16506	95534	16506	95534
439	=439			96580	480267			17322	96487	16535	95700	16535	95700
440	=2*2*2*5*11			26730	138475			17334	96571	16551	95788	16551	95788
441	=3*3*7*7			28224	150151			17382	96903	16608	96122	16608	96122
442	=2*13*17			41769	214122			17406	97071	16638	96294	16638	96294
443	=443			98346	489073			17430	97243	16669	96472	16669	96472
444	=2*2*3*37			37962	190193			17442	97331	16686	96566	16686	96566
445	=5*89			60075	299466			17466	97511	16719	96756	16719	96756
446	=2*223			74928	372409			17478	97603	16737	96856	16737	96856
447	=3*149			67050	333457			17490	97699	16756	96962	16756	96962
448	=2*2*2*2*2*7			20412	107955			17496	97749	16767	97020	16767	97020
449	=449			101025	502432			17688	98961	16960	98170	16960	98170
450	=2*3*3*5*5			24300	131813			17784	99569	17058	98750	17058	98750
451	=11*41			56826	285825			17880	100181	17157	99336	17157	99336
452	=2*2*113			57969	288256			17928	100489	17208	99634	17208	99634
453	=3*151			68856	342463			18024	101109	17309	100232	17309	100232
454	=2*227			77634	385899			18072	101421	17361	100536	17361	100536

455	=5*7*13			38220	197731			18120	101737	17414	100846	17414	100846
456	=2*2*2*3*19			30780	156221			18144	101897	17442	101006	17442	101006
457	=457			104653	520524			18240	102533	17547	101628	17547	101628
458	=2*229			79005	392734			18288	102853	17601	101944	17601	101944
459	=3*3*3*17			33048	168655			18336	103177	17656	102266	17656	102266
460	=2*2*5*23			37260	188643			18360	103341	17685	102432	17685	102432
461	=461			106491	529690			18408	103673	17742	102766	17742	102766
462	=2*3*7*11			33264	172937			18432	103841	17772	102938	17772	102938
463	=463			107416	534303			18456	104013	17803	103116	17803	103116
464	=2*2*2*2*29			35235	177082			18468	104101	17820	103210	17820	103210
465	=3*5*31			44640	224935			18564	104769	17933	103880	17933	103880
466	=2*233			81783	406584			18612	105105	17991	104220	17991	104220
467	=467			109278	543589			18660	105445	18050	104566	18050	104566
468	=2*2*3*3*13			29484	151715			18684	105617	18081	104744	18081	104744
469	=7*67			63784	318879			18732	105965	18142	105102	18142	105102
470	=2*5*47			50760	254209			18756	106141	18174	105286	18174	105286
471	=3*157			74418	370201			18780	106321	18207	105476	18207	105476
472	=2*2*2*59			47790	238335			18792	106413	18225	105576	18225	105576
473	=11*43			62436	313963			18840	106777	18290	105958	18290	105958
474	=2*3*79			56880	283433			18864	106961	18324	106154	18324	106154
475	=5*5*19			42750	218181			18888	107149	18359	106356	18359	106356
476	=2*2*7*17			38556	197155			18900	107245	18378	106462	18378	106462
477	=3*3*53			51516	257539			18924	107441	18415	106676	18415	106676
478	=2*239			86040	427809			18936	107541	18435	106788	18435	106788
479	=479			114960	571927			18948	107645	18456	106906	18456	106906
480	=2*2*2*2*2*3*5			21870	118391			18954	107699	18468	106970	18468	106970
481	=13*37			63973	322596			19050	108431	18597	107736	18597	107736
482	=2*241			87483	435004			19098	108799	18663	108124	18663	108124
483	=3*7*23			46368	235447			19146	109171	18730	108518	18730	108518
484	=2*2*11*11			39204	204219			19170	109359	18765	108720	18765	108720
485	97			71295	355486			19218	109739	18834	109126	18834	109126
486	=2*3*3*3*3*3			23328	131417			19242	109931	18870	109334	18870	109334
487	=487			118828	591219			19266	110127	18907	109548	18907	109548
488	=2*2*2*61			51057	254650			19278	110227	18927	109660	18927	109660
489	=3*163			80196	399019			19326	110623	19000	110090	19000	110090
490	=2*5*7*7			35280	188169			19350	110823	19038	110310	19038	110310
491	=491			120786	600985			19374	111027	19077	110536	19077	110536
492	=2*2*3*41			46494	232901			19386	111131	19098	110654	19098	110654
493	=17*29			66555	337434			19410	111343	19139	110892	19139	110892
494	=2*13*19			51870	265303			19422	111451	19161	111016	19161	111016
495	=3*3*5*11			35640	185359			19434	111563	19184	111146	19184	111146
496	=2*2*2*2*31			40176	201855			19440	111621	19197	111216	19197	111216
497	=7*71			71568	357799			19488	112049	19278	111694	19278	111694
498	=2*3*83			62748	312725			19512	112265	19320	111938	19320	111938
499	=499			124750	620757			19536	112485	19363	112188	19363	112188
500	=2*2*5*5*5			30375	165238			19548	112597	19386	112318	19386	112318
501	=3*167			84168	418831			19572	112825	19431	112580	19431	112580
502	=2*251			94878	471879			19584	112941	19455	112716	19455	112716
503	=503			126756	630763			19596	113061	19480	112858	19480	112858
504	=2*2*2*3*3*7			27216	144497			19602	113123	19494	112934	19494	112934
505	=5*101			77265	385296			19626	113367	19543	113220	19543	113220
506	=2*11*23			54648	277969			19638	113491	19569	113368	19569	113368
507	=3*13*13			49686	257629			19650	113619	19596	113522	19596	113522
508	=2*2*127			73152	363975			19656	113685	19611	113604	19611	113604
509	=509			129795	645922			19668	113821	19640	113770	19640	113770
510	=2*3*5*17			41310	211271			19674	113891	19656	113858	19656	113858
511	=7*73			75628	378099			19680	113965	19673	113952	19673	113952
512	=2*2*2*2*2*2*2*2*2			19683	114004			19683	114004	19683	114004	19683	114004
513	=3*3*3*19			41040	209047			20451	117328	20196	117074	20196	117074
514	=2*257			99459	494724			20835	118992	20454	118614	20454	118614
515	=5*103			80340	400651			21219	120660	20713	120160	20713	120160
516	=2*2*3*43			51084	255875			21411	121496	20844	120938	20844	120938
517	=11*47			74448	374199			21795	123172	21105	122496	21105	122496
518	=2*7*37			59052	297205			21987	124012	21237	123280	21237	123280
519	=3*173			90306	449449			22179	124856	21370	124070	21370	124070
520	=2*2*2*5*13			36855	190080			22275	125280	21438	124470	21438	124470
521	=521			135981	676780			22659	126972	21703	126052	21703	126052
522	=2*3*3*29			46980	236861			22851	127820	21837	126848	21837	126848
523	=523			137026	681993			23043	128672	21972	127650	21972	127650
524	=2*2*131			77814	387229			23139	129100	22041	128056	22041	128056
525	=3*5*5*7			37800	201775			23331	129960	22178	128870	22178	128870
526	=2*263			104148	518109			23427	130392	22248	129282	22248	129282
527	=17*31			75888	384439			23523	130828	22319	129700	22319	129700
528	=2*2*2*2*3*11			32076	166283			23571	131048	22356	129914	22356	129914
529	=23*23			76176	388839			23955	132772	22629	131544	22629	131544
530	=2*5*53			64395	322444			24147	133636	22767	132364	22767	132364
531	=3*3*59			63720	318559			24339	134504	22906	133190	22906	133190
532	=2*2*7*19			47880	244335			24435	134940	22977	133608	22977	133608

533	=13*41			78351	394798			24627	135816	23118	134446	23118	134446
534	=2*3*89			72090	359363			24723	136256	23190	134870	23190	134870
535	=5*107			86670	432261			24819	136700	23263	135300	23263	135300
536	=2*2*2*67			61506	306835			24867	136924	23301	135520	23301	135520
537	=3*179			96660	481147			25059	137816	23446	136382	23446	136382
538	=2*269			108945	542034			25155	138264	23520	136818	23520	136818
539	=7*7*11			51744	270439			25251	138716	23595	137260	23595	137260
540	=2*2*3*3*3*5			29160	158591			25299	138944	23634	137486	23634	137486
541	=541			146611	729810			25395	139404	23711	137940	23711	137940
542	=2*271			110568	550129			25443	139636	23751	138172	23751	138172
543	=3*181			98826	491953			25491	139872	23792	138410	23792	138410
544	=2*2*2*2*2*17			37179	189564			25515	139992	23814	138534	23814	138534
545	=5*109			89925	448516			25899	141780	24103	140260	24103	140260
546	=2*3*7*13			45864	237281			26091	142676	24249	141128	24249	141128
547	=547			149878	746109			26283	143576	24396	142002	24396	142002
548	=2*2*137			85077	423460			26379	144028	24471	142444	24471	142444
549	=3*3*61			68076	340339			26571	144936	24620	143330	24620	143330
550	=2*5*5*11			44550	232239			26667	145392	24696	143778	24696	143778
551	=19*29			82650	419521			26763	145852	24773	144232	24773	144232
552	=2*2*2*3*23			44712	226409			26811	146084	24813	144464	24813	144464
553	=7*79			88480	442359			27003	147008	24966	145374	24966	145374
554	=2*277			115509	574774			27099	147472	25044	145834	25044	145834
555	=3*5*37			63270	318445			27195	147940	25123	146300	25123	146300
556	=2*2*139			87570	435897			27243	148176	25164	146538	25164	146538
557	=557			155403	773674			27339	148652	25245	147016	25245	147016
558	=2*3*3*31			53568	269945			27387	148892	25287	147260	25287	147260
559	=13*43			86086	433629			27435	149136	25330	147510	25330	147510
560	=2*2*2*2*5*7			34020	180931			27459	149260	25353	147640	25353	147640
561	=3*11*17			60588	311251			27651	150216	25514	148598	25514	148598
562	=2*281			118863	591504			27747	150696	25596	149082	25596	149082
563	=563			158766	790453			27843	151180	25679	149572	25679	149572
564	=2*2*3*47			60912	305063			27891	151424	25722	149822	25722	149822
565	=5*113			96615	481926			27987	151916	25807	150324	25807	150324
566	=2*283			120558	599959			28035	152164	25851	150580	25851	150580
567	=3*3*3*3*7			36288	193495			28083	152416	25896	150842	25896	150842
568	=2*2*2*71			69012	344325			28107	152544	25920	150978	25920	150978
569	=569			162165	807412			28203	153052	26009	151504	26009	151504
570	=2*3*5*19			51300	261821			28251	153308	26055	151772	26055	151772
571	=571			163306	813105			28299	153568	26102	152046	26102	152046
572	=2*2*11*13			54054	280141			28323	153700	26127	152188	26127	152188
573	=3*191			110016	547783			28371	153968	26176	152474	26176	152474
574	=2*7*41			72324	363789			28395	154104	26202	152622	26202	152622
575	=5*5*23			62100	315931			28419	154244	26229	152776	26229	152776
576	=2*2*2*2*2*2*3*3			26244	147755			28431	154316	26244	152858	26244	147755
577	=577			166753	830304			28815	156232	26565	154776	26565	154776
578	=2*17*17			70227	361300			29007	157192	26727	155740	26727	155740
579	=3*193			112326	559309			29199	158156	26890	156710	26890	156710
580	=2*2*5*29			58725	296628			29295	158640	26973	157200	26973	157200
581	=7*83			97608	487999			29487	159612	27138	158182	27138	158182
582	=2*3*97			85554	426587			29583	160100	27222	158678	27222	158678
583	=11*53			94446	474453			29679	160592	27307	159180	27307	159180
584	=2*2*2*73			72927	363880			29727	160840	27351	159436	27351	159436
585	=3*3*5*13			49140	254299			29919	161828	27520	160442	27520	160442
586	=2*293			129213	643134			30015	162324	27606	160950	27606	160950
587	=587			172578	859369			30111	162824	27693	161464	27693	161464
588	=2*2*3*7*7			42336	225815			30159	163076	27738	161726	27738	161726
589	=19*31			94240	477927			30255	163584	27827	162252	27827	162252
590	=2*5*59			79650	398779			30303	163840	27873	162520	27873	162520
591	=3*197			117018	582721			30351	164100	27920	162794	27920	162794
592	=2*2*2*2*37			56943	285894			30375	164232	27945	162936	27945	162936
593	=593			176121	877048			30567	165252	28122	163990	28122	163990
594	=2*3*3*3*11			42768	222569			30663	165764	28212	164522	28212	164522
595	=5*7*17			64260	330171			30759	166280	28303	165060	28303	165060
596	=2*2*149			100575	500782			30807	166540	28350	165334	28350	165334
597	=3*199			119400	594607			30903	167064	28443	165884	28443	165884
598	=2*13*23			75348	384045			30951	167328	28491	166164	28491	166164
599	=599			179700	894907			30999	167596	28540	166450	28540	166450
600	=2*2*2*3*5*5			36450	198323			31023	167732	28566	166598	28566	166598

Anhang 4

Gatter-Komplexität der optimalen Kombinationen der MM

Tabelle 13 zeigt die Gatter-Komplexität der folgenden MM für alle Polynom-Längen bis 600 Bits:

- die von [30] rekonstruierten Werte für die optimale Kombination von 6 MM (siehe Kapitel 5.2.1)
- die im Rahmen dieser Arbeit mit Algorithmus 1 (siehe Kapitel 3.5) ermittelten Werte für die optimale Kombination von 9 MM
- die mittels Algorithmus 7 (siehe Kapitel 5.2.2) verbesserten Ergebnisse des Algorithmus 1
- die im Rahmen dieser Arbeit mit Algorithmus 1 ermittelten Werte für die optimale Kombination von 10 MM, also unter der Berücksichtigung einer weiterer Verbesserungs-Technik, die als io_uK-MM zur Menge der untersuchten MM hinzugefügt wurde (siehe Kapitel 5.2.2).

Tabelle 13: Gatter-Komplexität der optimalen Kombinationen der MM

n	Rekonstruierte Daten aus [30]		Ergebnisse dieser Arbeit					
			Optimale Kombination der 9 MM nach Algorithmus 1		Optimale Kombination der 9 MM nach Algorithmus 7		Optimale Kombination der 10 MM nach Algorithmus 7	
	#AND	#XOR	#AND	#XOR	#AND	#XOR	#AND	#XOR
1	1	0	1	0	1	0	1	0
2	4	1	4	1	4	1	4	1
3	9	4	9	4	9	4	9	4
4	16	9	16	9	16	9	16	9
5	25	16	25	16	25	16	25	16
6	27	32	27	30	27	30	27	30
7	49	36	49	36	49	36	49	36
8	48	55	48	52	48	52	48	52
9	54	80	54	72	54	72	54	72
10	75	84	75	80	75	80	75	80

11	121	100	121	100	121	100	78	110
12	81	140	81	127	81	127	81	127
13	147	160	147	154	147	154	124	143
14	147	160	147	154	147	154	147	154
15	150	196	150	180	150	180	150	180
16	144	225	144	206	144	206	144	206
17	162	308	162	269	162	269	155	251
18	162	308	162	269	162	269	162	269
19	225	328	225	303	225	303	203	294
20	225	328	225	303	225	303	225	303
21	294	360	294	336	294	336	230	369
22	363	384	363	374	363	374	234	404
23	243	512	288	431	288	431	239	440
24	243	512	288	431	288	431	288	431
25	325	638	350	486	350	486	328	496
26	324	668	324	574	324	574	372	517
27	324	668	324	574	324	574	324	574
28	441	588	441	551	441	551	441	551
29	450	704	450	629	450	629	446	611
30	450	704	450	629	450	629	450	629
31	432	799	432	727	432	727	437	696
32	432	799	432	727	432	727	432	727
33	486	1064	406	938	425	912	453	819
34	486	1064	486	930	486	923	465	869
35	486	1064	486	930	486	930	478	907
36	486	1064	486	930	486	930	486	930
37	675	1140	675	1046	675	1039	567	982
38	675	1140	675	1046	675	1039	609	1012
39	675	1140	675	1046	675	1046	652	1032
40	675	1140	675	1046	675	1046	675	1046
41	882	1244	882	1133	882	1133	684	1180
42	882	1244	882	1133	882	1133	882	1133
43	900	1496	900	1318	900	1318	697	1323
44	900	1496	900	1318	900	1318	702	1353
45	900	1496	900	1318	900	1318	900	1318
46	729	1724	864	1458	864	1451	717	1478
47	729	1724	864	1458	864	1458	814	1462
48	729	1724	864	1458	864	1458	864	1458
49	975	1992	1050	1630	1050	1630	943	1590
50	975	1992	1050	1630	1050	1630	984	1660
51	972	2216	972	1908	972	1901	1071	1704
52	972	2216	972	1908	972	1901	1116	1718
53	972	2216	972	1908	972	1908	1019	1846
54	972	2216	972	1908	972	1908	972	1908
55	1323	1984	1323	1846	1323	1846	1205	1864
56	1323	1984	1323	1846	1323	1846	1323	1846
57	1350	2348	1350	2094	1350	2087	1332	1968
58	1350	2348	1350	2094	1350	2087	1338	2033
59	1350	2348	1350	2094	1350	2094	1345	2071
60	1350	2348	1350	2094	1350	2094	1350	2094
61	1296	2649	1296	2387	1296	2387	1323	2230
62	1296	2649	1296	2387	1296	2387	1311	2302
63	1296	2649	1296	2387	1296	2387	1300	2366
64	1296	2649	1296	2387	1296	2387	1296	2387
65	2178	2756	1196	2872	1234	2820	1337	2588
66	2178	2756	1218	3018	1275	2940	1359	2685
67	1458	3476	1264	3131	1359	3010	1382	2787
68	1458	3476	1458	3022	1458	2988	1395	2826
69	1458	3476	1458	3022	1458	3022	1420	2920
70	1458	3476	1458	3022	1458	3022	1434	2963
71	1458	3476	1458	3022	1458	3022	1449	3011
72	1458	3476	1458	3022	1458	3022	1458	3022
73	2025	3736	1950	3322	1731	3413	1619	3145
74	2025	3736	1950	3322	1950	3322	1701	3202
75	2025	3736	1950	3322	1950	3322	1784	3264
76	2025	3736	2025	3396	2025	3362	1827	3281
77	2025	3736	2025	3396	2025	3396	1912	3341
78	2025	3736	2025	3396	2025	3396	1956	3366
79	2025	3736	2025	3396	2025	3396	2001	3396
80	2025	3736	2025	3396	2025	3396	2025	3396
81	1944	4592	1944	3924	1944	3924	2042	3685
82	2646	4064	2646	3690	2646	3683	2052	3824
83	2646	4064	2646	3690	2646	3690	2447	3732
84	2646	4064	2646	3690	2646	3690	2070	4024
85	2700	4844	2700	4266	2700	4252	2275	4072
86	2700	4844	2700	4266	2700	4252	2091	4267
87	2700	4844	2700	4266	2700	4259	2100	4329

88	2700	4844	2700	4266	2700	4259	2106	4364
89	2700	4844	2700	4266	2700	4266	2501	4296
90	2700	4844	2700	4266	2700	4266	2700	4266
91	2187	5552	2592	4684	2592	4650	2333	4588
92	2187	5552	2592	4684	2592	4650	2151	4731
93	2187	5552	2592	4684	2592	4684	2344	4723
94	2187	5552	2592	4684	2592	4684	2442	4712
95	2187	5552	2592	4684	2592	4684	2541	4706
96	2187	5552	2592	4684	2592	4684	2592	4684
97	2925	6104	2925	5256	2663	5374	2749	4973
98	2925	6104	2925	5256	2925	5256	2829	5110
99	2925	6104	2925	5256	2925	5256	2910	5252
100	2925	6104	2925	5256	2925	5256	2925	5256
101	2916	7076	2916	6073	2916	6039	3125	5417
102	2916	7076	2916	6073	2916	6039	3213	5466
103	2916	7076	2916	6073	2916	6039	3302	5496
104	2916	7076	2916	6073	2916	6039	3348	5515
105	2916	7076	2916	6073	2916	6073	3153	5773
106	2916	7076	2916	6073	2916	6073	3057	5906
107	2916	7076	2916	6073	2916	6073	2962	6032
108	2916	7076	2916	6073	2916	6073	2916	6073
109	3969	6396	3969	5900	3969	5900	3381	6013
110	3969	6396	3969	5900	3969	5900	3615	5974
111	3969	6396	3969	5900	3969	5900	3850	5940
112	3969	6396	3969	5900	3969	5900	3969	5900
113	4050	7520	4050	6670	3739	6762	3986	6173
114	4050	7520	4050	6670	4050	6636	3996	6300
115	4050	7520	4050	6670	4050	6636	4007	6432
116	4050	7520	4050	6670	4050	6636	4014	6474
117	4050	7520	4050	6670	4050	6670	4027	6580
118	4050	7520	4050	6670	4050	6670	4035	6623
119	4050	7520	4050	6670	4050	6670	4044	6671
120	4050	7520	4050	6670	4050	6670	4050	6670
121	3888	8455	3888	7606	3704	7690	3995	6973
122	3888	8455	3888	7606	3888	7585	3969	7114
123	3888	8455	3888	7606	3888	7586	3944	7260
124	3888	8455	3888	7606	3888	7586	3933	7307
125	3888	8455	3888	7606	3888	7599	3910	7467
126	3888	8455	3888	7606	3888	7599	3900	7536
127	3888	8455	3888	7606	3888	7606	3891	7580
128	3888	8455	3888	7606	3888	7606	3888	7606
129	6534	8792	3592	8653	3668	8549	3969	8010
130	6534	8792	3588	9013	3702	8857	4011	8216
131	6534	8792	3632	9310	3784	9102	4054	8412
132	6534	8792	3654	9459	3825	9225	4077	8482
133	5400	9956	4212	9248	3993	9370	4122	8720
134	5400	9956	4212	9248	4212	9248	4146	8827
135	5400	9956	4212	9248	4212	9248	4171	8907
136	4374	11000	4374	9567	4374	9437	4185	8951
137	4374	11000	4374	9567	4239	9626	4234	9141
138	4374	11000	4374	9567	4374	9546	4260	9240
139	4374	11000	4374	9567	4374	9549	4287	9328
140	4374	11000	4374	9567	4374	9549	4302	9342
141	4374	11000	4374	9567	4374	9560	4331	9474
142	4374	11000	4374	9567	4374	9560	4347	9527
143	4374	11000	4374	9567	4374	9567	4364	9551
144	4374	11000	4374	9567	4374	9567	4374	9567
145	5850	11932	4566	10866	4912	10498	4695	9815
146	5850	11932	5850	10164	5850	10164	4857	9943
147	5850	11932	5850	10164	5850	10164	5020	10059
148	5850	11932	5850	10164	5850	10164	5103	10085
149	5850	11932	5850	10164	5850	10164	5268	10247
150	5850	11932	5850	10164	5850	10164	5352	10314
151	6075	11844	6075	10745	6075	10615	5437	10350
152	6075	11844	6075	10745	6075	10615	5481	10372
153	6075	11844	6075	10745	6075	10724	5650	10494
154	6075	11844	6075	10745	6075	10724	5736	10559
155	6075	11844	6075	10745	6075	10729	5823	10611
156	6075	11844	6075	10745	6075	10729	5868	10603
157	6075	11844	6075	10745	6075	10738	5957	10703
158	6075	11844	6075	10745	6075	10738	6003	10738
159	6075	11844	6075	10745	6075	10745	6050	10740
160	6075	11844	6075	10745	6075	10745	6075	10745
161	5832	14420	5954	12149	6097	11987	6108	11325
162	5832	14420	5832	12257	5832	12257	6126	11619
163	7938	12860	7938	11614	7938	11580	6145	11899
164	7938	12860	7938	11614	7938	11580	6156	12003
165	7938	12860	7938	11614	7938	11614	6945	11861

166	7938	12860	7938	11614	7938	11614	7341	11774
167	7938	12860	7938	11614	7938	11614	6586	12360
168	7938	12860	7938	11614	7938	11614	7938	11614
169	8100	15248	7364	13719	7716	13286	6619	12755
170	8100	15248	8100	13381	6318	14254	6825	12808
171	8100	15248	8100	13381	8100	13313	6456	13200
172	8100	15248	8100	13381	8100	13313	6273	13358
173	8100	15248	8100	13381	6318	14412	6290	13526
174	8100	15248	8100	13381	6318	14412	6300	13593
175	8100	15248	8100	13381	6318	14412	6311	13665
176	8100	15248	8100	13381	8100	13347	6318	13662
177	8100	15248	8100	13381	8100	13381	7107	13571
178	8100	15248	8100	13381	8100	13381	7503	13508
179	8100	15248	8100	13381	8100	13381	7900	13450
180	8100	15248	8100	13381	8100	13381	8100	13381
181	6561	17420	7776	14721	7776	14584	7365	14071
182	6561	17420	7776	14721	7776	14584	6999	14398
183	6561	17420	7776	14721	7776	14591	6634	14686
184	6561	17420	7776	14721	7776	14591	6453	14834
185	6561	17420	7776	14721	7776	14700	6838	14820
186	6561	17420	7776	14721	7776	14700	7032	14817
187	6561	17420	7776	14721	7776	14707	7227	14797
188	6561	17420	7776	14721	7776	14707	7326	14745
189	6561	17420	7776	14721	7776	14714	7523	14781
190	6561	17420	7776	14721	7776	14714	7623	14780
191	6561	17420	7776	14721	7776	14721	7724	14738
192	6561	17420	7776	14721	7776	14721	7776	14721
193	8775	18540	7840	16376	7642	16412	8089	15301
194	8775	18540	8775	16255	8775	16188	8247	15595
195	8775	18540	8775	16255	8775	16188	8406	15871
196	8775	18540	8775	16255	8775	16255	8487	15965
197	8775	18540	8775	16255	8775	16255	8648	16299
198	8775	18540	8775	16255	8775	16255	8775	16255
199	8775	18540	8775	16255	8775	16255	8775	16255
200	8775	18540	8775	16255	8775	16255	8775	16255
201	8748	22088	8900	18684	8889	18326	9174	16789
202	8748	22088	10350	17998	9900	17918	9375	16955
203	8748	22088	10350	17998	9900	17918	9550	17055
204	8748	22088	10350	17998	9900	18061	9639	17059
205	8748	22088	10350	17998	9900	18061	9816	17171
206	8748	22088	10350	17998	9900	18061	9906	17206
207	8748	22088	10350	17998	9900	18061	9997	17246
208	8748	22088	10350	17998	9900	18061	10044	17219
209	8748	22088	10350	17998	9900	18061	9653	17788
210	8748	22088	10350	17998	9900	18061	9459	18051
211	8748	22088	8748	18972	8748	18958	9266	18319
212	8748	22088	8748	18972	8748	18958	9171	18405
213	8748	22088	8748	18972	8748	18965	8980	18711
214	8748	22088	8748	18972	8748	18965	8886	18842
215	8748	22088	8748	18972	8748	18972	8793	18926
216	8748	22088	8748	18972	8748	18972	8748	18972
217	11907	20080	11907	18481	11907	18460	9677	18854
218	11907	20080	11907	18481	11907	18460	10143	18799
219	11907	20080	11907	18481	11907	18467	10610	18723
220	11907	20080	11907	18481	11907	18467	10845	18635
221	11907	20080	11907	18481	11907	18474	11314	18623
222	11907	20080	11907	18481	11907	18474	11550	18594
223	11907	20080	11907	18481	11907	18481	11787	18516
224	11907	20080	11907	18481	11907	18481	11907	18481
225	11700	23638	11700	20126	11700	20126	11940	19029
226	12150	23516	12150	20847	11217	20973	11958	19307
227	12150	23516	12150	20847	11543	21019	11977	19563
228	12150	23516	12150	20847	12150	20703	11988	19639
229	12150	23516	12150	20847	12150	20710	12009	19961
230	12150	23516	12150	20847	12150	20710	12021	20098
231	12150	23516	12150	20847	12150	20717	12034	20184
232	12150	23516	12150	20847	12150	20717	12042	20231
233	12150	23516	12150	20847	12150	20826	12067	20445
234	12150	23516	12150	20847	12150	20826	12081	20556
235	12150	23516	12150	20847	12150	20833	12105	20634
236	12150	23516	12150	20847	12150	20833	12105	20634
237	12150	23516	12150	20847	12150	20840	12122	20790
238	12150	23516	12150	20847	12150	20840	12132	20843
239	12150	23516	12150	20847	12150	20847	12143	20843
240	12150	23516	12150	20847	12150	20847	12150	20847
241	11664	26385	10774	24062	11186	23340	12039	21455
242	11664	26385	11664	23648	11112	23805	11985	21763
243	11664	26385	11664	23648	11664	23546	11932	22047

244	11664	26385	11664	23648	11664	23546	11907	22133
245	11664	26385	11664	23648	11664	23580	11856	22487
246	11664	26385	11664	23648	11664	23580	11832	22638
247	11664	26385	11664	23648	11664	23580	11809	22734
248	11664	26385	11664	23648	11664	23580	11799	22786
249	11664	26385	11664	23648	11664	23614	11752	23108
250	11664	26385	11664	23648	11664	23614	11730	23273
251	11664	26385	11664	23648	11664	23614	11709	23413
252	11664	26385	11664	23648	11664	23614	11700	23425
253	11664	26385	11664	23648	11664	23648	11681	23577
254	11664	26385	11664	23648	11664	23648	11673	23626
255	11664	26385	11664	23648	11664	23648	11664	23648
256	11664	26385	11664	23648	11664	23648	11664	23648
257	12636	33844	10880	25893	11032	25685	11825	24521
258	12636	33844	10776	26741	11004	26429	11907	24930
259	12636	33844	10768	27466	11072	27050	11990	25344
260	12636	33844	10764	27829	11106	27361	12033	25491
261	12636	33844	10852	28428	11270	27856	12118	25949
262	12636	33844	10896	28729	11352	28105	12162	26150
263	12636	33844	10940	29036	11434	28360	12207	26292
264	12636	33844	10962	29179	11475	28477	12231	26367
265	12636	33844	12312	28544	11811	28772	12320	26845
266	12636	33844	12312	28544	12312	28544	12366	27088
267	12636	33844	12312	28544	12312	28544	12413	27304
268	12636	33844	12312	28544	12312	28544	12438	27350
269	12636	33844	12312	28544	12312	28544	12487	27578
270	12636	33844	12312	28544	12312	28544	12513	27663
271	13122	34148	13122	29635	12795	29308	12555	27735
272	13122	34148	13122	29635	12818	29341	12555	27735
273	13122	34148	13122	29635	12700	29622	12652	28184
274	13122	34148	13122	29635	12717	29713	12702	28379
275	13122	34148	13122	29635	13122	29533	12753	28579
276	13122	34148	13122	29635	13122	29533	12780	28615
277	13122	34148	13122	29635	13122	29567	12833	28861
278	13122	34148	13122	29635	13122	29567	12861	28954
279	13122	34148	13122	29635	13122	29567	12890	28984
280	13122	34148	13122	29635	13122	29567	12906	29003
281	13122	34148	13122	29635	13122	29601	12963	29269
282	13122	34148	13122	29635	13122	29601	12993	29406
283	13122	34148	13122	29635	13122	29601	13041	29502
284	13122	34148	13122	29635	13122	29601	13041	29502
285	13122	34148	13122	29635	13122	29635	13074	29622
286	13122	34148	13122	29635	13122	29635	13092	29651
287	13122	34148	13122	29635	13122	29635	13122	29635
288	13122	34148	13122	29635	13122	29635	13122	29635
289	17550	36124	13490	32526	14182	31790	13763	30204
290	17550	36124	13698	33481	14736	32377	14085	30457
291	17550	36124	17550	31219	17550	31152	14408	30715
292	17550	36124	17550	31219	17550	31152	14571	30776
293	17550	36124	17550	31219	17550	31152	14896	31082
294	17550	36124	17550	31219	17550	31152	15060	31203
295	17550	36124	17550	31219	17550	31152	15225	31257
296	17550	36124	17550	31219	17550	31219	15309	31288
297	17550	36124	17550	31219	17550	31219	15638	31614
298	17550	36124	17550	31219	17550	31219	15804	31781
299	17550	36124	17550	31219	17550	31219	15971	31917
300	17550	36124	17550	31219	17550	31219	16056	31915
301	18225	36808	18225	33273	18225	32831	16225	32063
302	18225	36808	18225	33273	18225	32831	16311	32104
303	18225	36808	18225	33273	18225	32831	16443	32102
304	18225	36808	18225	33273	18225	32831	16443	32102
305	18225	36808	18225	33273	18225	33171	16780	32423
306	18225	36808	18225	33273	18225	33171	16950	32550
307	18225	36808	18225	33273	18225	33171	17121	32682
308	18225	36808	18225	33273	18225	33171	17208	32676
309	18225	36808	18225	33273	18225	33205	17381	32858
310	18225	36808	18225	33273	18225	33205	17469	32915
311	18225	36808	18225	33273	18225	33205	17558	32901
312	18225	36808	18225	33273	18225	33205	17604	32898
313	18225	36808	18225	33273	18225	33239	17781	33100
314	18225	36808	18225	33273	18225	33239	17871	33205
315	18225	36808	18225	33273	18225	33239	18009	33239
316	18225	36808	18225	33273	18225	33239	18009	33239
317	18225	36808	18225	33273	18225	33273	18102	33323
318	18225	36808	18225	33273	18225	33273	18225	33273
319	18225	36808	18225	33273	18225	33273	18225	33273
320	18225	36808	18225	33273	18225	33273	18225	33273
321	17496	44552	17759	36538	18045	36214	18290	34514

322	17496	44552	17862	37426	18291	36940	18324	35099
323	17496	44552	17496	37902	17496	37902	18359	35689
324	17496	44552	17496	37902	17496	37902	18378	35908
325	23814	39920	23814	36015	22430	36674	18415	36550
326	23814	39920	23814	36015	22430	36674	18435	36835
327	23814	39920	23814	36015	19008	38730	18456	37045
328	23814	39920	23814	36015	19008	38730	18468	37154
329	23814	39920	23814	36015	19008	38730	20045	36872
330	23814	39920	23814	36015	19008	38873	20835	36735
331	23814	39920	23814	36015	19008	38873	21626	36563
332	23814	39920	23814	36015	19008	38873	22023	36399
333	23814	39920	23814	36015	19008	38873	20512	37655
334	23814	39920	23814	36015	19008	38873	19758	38246
335	23814	39920	23814	36015	19008	38873	22461	36756
336	23814	39920	23814	36015	19008	38873	18630	39061
337	18954	51916	22092	42189	18954	43377	21175	38299
338	18954	51916	22092	42189	18954	43377	19857	39445
339	18954	51916	18954	44087	18954	43377	20268	39553
340	18954	51916	18954	44087	18954	43377	20475	39527
341	18954	51916	18954	44087	18954	43886	19736	40397
342	18954	51916	18954	44087	18954	43886	19368	40794
343	18954	51916	18954	44087	18954	43886	19001	41112
344	18954	51916	18954	44087	18954	43886	18819	41275
345	18954	51916	18954	44087	18954	43886	18852	41613
346	18954	51916	18954	44087	18954	43953	18870	41786
347	18954	51916	18954	44087	18954	43953	18900	41908
348	18954	51916	18954	44087	18954	43953	18900	41908
349	18954	51916	18954	44087	18954	43953	18921	42140
350	18954	51916	18954	44087	18954	43953	18933	42217
351	18954	51916	18954	44087	18954	44020	18946	42213
352	18954	51916	18954	44087	18954	44020	18954	42215
353	18954	51916	18954	44087	18954	44020	20531	42035
354	18954	51916	18954	44087	18954	44020	21321	41949
355	18954	51916	18954	44087	18954	44020	22112	41825
356	18954	51916	18954	44087	18954	44087	22509	41679
357	18954	51916	18954	44087	18954	44087	18954	44087
358	18954	51916	18954	44087	18954	44087	18954	44087
359	18954	51916	18954	44087	18954	44087	18954	44087
360	18954	51916	18954	44087	18954	44087	18954	44087
361	19683	53792	23328	45409	23328	44933	22829	42782
362	19683	53792	23328	45409	23328	44933	22095	43477
363	19683	53792	23328	45409	23328	44933	21362	44133
364	19683	53792	23328	45409	23328	44933	20997	44375
365	19683	53792	23328	45409	23328	44967	20266	45043
366	19683	53792	23328	45409	23328	44967	19902	45336
367	19683	53792	23328	45409	23328	44967	19359	45696
368	19683	53792	23328	45409	23328	44967	19359	45696
369	19683	53792	23328	45409	23328	45307	20128	45761
370	19683	53792	23328	45409	23328	45307	20514	45752
371	19683	53792	23328	45409	23328	45307	20901	45748
372	19683	53792	23328	45409	23328	45307	21096	45658
373	19683	53792	23328	45409	23328	45341	21485	45712
374	19683	53792	23328	45409	23328	45341	21681	45697
375	19683	53792	23328	45409	23328	45341	21878	45595
376	19683	53792	23328	45409	23328	45341	21978	45548
377	19683	53792	23328	45409	23328	45375	22371	45622
378	19683	53792	23328	45409	23328	45375	22569	45663
379	19683	53792	23328	45409	23328	45375	22869	45573
380	19683	53792	23328	45409	23328	45375	22869	45573
381	19683	53792	23328	45409	23328	45409	23328	45409
382	19683	53792	23328	45409	23328	45409	23328	45409
383	19683	53792	23328	45409	23328	45409	23328	45409
384	19683	53792	23328	45409	23328	45409	23328	45409
385	26325	56048	23520	50296	22926	50404	23953	46666
386	26325	56048	23520	50296	22926	50404	24267	47251
387	26325	56048	26325	49485	26325	49351	24582	47841
388	26325	56048	26325	49485	26325	49351	24741	48044
389	26325	56048	26325	49485	26325	49351	25058	48694
390	26325	56048	26325	49485	26325	49351	25218	48975
391	26325	56048	26325	49485	26325	49418	25379	49165
392	26325	56048	26325	49485	26325	49418	25461	49264
393	26325	56048	26325	49485	26325	49418	26013	49418
394	26325	56048	26325	49485	26325	49418	26013	49418
395	26325	56048	26325	49485	26325	49418	26013	49418
396	26325	56048	26325	49485	26325	49485	26325	49485
397	26325	56048	26325	49485	26325	49485	26325	49485
398	26325	56048	26325	49485	26325	49485	26325	49485
399	26325	56048	26325	49485	26325	49485	26325	49485

400	26325	56048	26325	49485	26325	49485	26325	49485
401	25272	67270	24624	55846	24624	55846	27122	51232
402	25272	67270	24624	55846	24624	55846	27522	51771
403	25272	67270	24624	55846	24624	55846	27923	52105
404	25272	67270	24624	55846	24624	55846	28125	52176
405	25272	67270	24624	55846	24624	55846	26546	53309
406	26244	67988	31050	55263	29700	54141	28650	52583
407	26244	67988	31050	55263	29700	54438	28827	52593
408	26244	67988	31050	55263	29700	54438	28917	52602
409	26244	67988	31050	55263	29700	54438	29270	52828
410	26244	67988	31050	55263	29700	54438	29448	52945
411	26244	67988	31050	55263	29700	54438	29718	52955
412	26244	67988	31050	55263	29700	54438	29718	52955
413	26244	67988	31050	55263	29700	54438	29899	53139
414	26244	67988	31050	55263	29700	54581	29991	53184
415	26244	67988	31050	55263	29700	54581	30084	53132
416	26244	67988	31050	55263	29700	54581	30132	53110
417	26244	67988	31050	55263	29700	54581	29349	54250
418	26244	67988	31050	55263	29700	54581	28959	54824
419	26244	67988	31050	55263	29700	54581	29700	54581
420	26244	67988	31050	55263	29700	54581	29700	54581
421	26244	67988	26244	58318	26244	58250	27990	56158
422	26244	67988	26244	58318	26244	58250	27798	56431
423	26244	67988	26244	58318	26244	58250	27607	56605
424	26244	67988	26244	58318	26244	58250	27513	56696
425	26244	67988	26244	58318	26244	58284	27130	57310
426	26244	67988	26244	58318	26244	58284	26940	57621
427	26244	67988	26244	58318	26244	58284	26658	57915
428	26244	67988	26244	58318	26244	58284	26658	57915
429	26244	67988	26244	58318	26244	58318	26244	58318
430	26244	67988	26244	58318	26244	58318	26244	58318
431	26244	67988	26244	58318	26244	58318	26244	58318
432	26244	67988	26244	58318	26244	58318	26244	58318
433	35721	62028	35721	56897	33075	58372	28101	58191
434	35721	62028	35721	56897	33075	58372	29031	58078
435	35721	62028	35721	56897	35721	56795	29962	57970
436	35721	62028	35721	56897	35721	56795	30429	57812
437	35721	62028	35721	56897	35721	56829	31362	57770
438	35721	62028	35721	56897	35721	56829	31830	57699
439	35721	62028	35721	56897	35721	56829	32299	57525
440	35721	62028	35721	56897	35721	56829	32535	57442
441	35721	62028	35721	56897	35721	56863	33472	57420
442	35721	62028	35721	56897	35721	56863	33942	57413
443	35721	62028	35721	56897	35721	56863	34413	57357
444	35721	62028	35721	56897	35721	56863	34650	57223
445	35721	62028	35721	56897	35721	56897	35123	57179
446	35721	62028	35721	56897	35721	56897	35361	57106
447	35721	62028	35721	56897	35721	56897	35721	56897
448	35721	62028	35721	56897	35721	56897	35721	56897
449	35100	71392	34200	61400	34200	61400	35786	58106
450	35100	71392	34200	61400	34200	61400	35820	58659
451	36450	72464	36450	64099	33651	64285	35855	59217
452	36450	72464	36450	64099	33651	64285	35874	59388
453	36450	72464	36450	64099	34629	64431	35911	60014
454	36450	72464	36450	64099	34629	64431	35931	60275
455	36450	72464	36450	64099	36450	63589	35952	60429
456	36450	72464	36450	64099	36450	63589	35964	60510
457	36450	72464	36450	64099	36450	63623	36005	61156
458	36450	72464	36450	64099	36450	63623	36027	61483
459	36450	72464	36450	64099	36450	63623	36050	61759
460	36450	72464	36450	64099	36450	63623	36063	61787
461	36450	72464	36450	64099	36450	63657	36088	62075
462	36450	72464	36450	64099	36450	63657	36102	62166
463	36450	72464	36450	64099	36450	63657	36126	62199
464	36450	72464	36450	64099	36450	63657	36126	62199
465	36450	72464	36450	64099	36450	63997	36175	62744
466	36450	72464	36450	64099	36450	63997	36201	62963
467	36450	72464	36450	64099	36450	63997	36228	63187
468	36450	72464	36450	64099	36450	63997	36243	63187
469	36450	72464	36450	64099	36450	64031	36290	63461
470	36450	72464	36450	64099	36450	64031	36315	63544
471	36450	72464	36450	64099	36450	64031	36314	63546
472	36450	72464	36450	64099	36450	64031	36315	63551
473	36450	72464	36450	64099	36450	64065	36348	63865
474	36450	72464	36450	64099	36450	64065	36366	64026
475	36450	72464	36450	64099	36450	64065	36396	64074
476	36450	72464	36450	64099	36450	64065	36396	64074
477	36450	72464	36450	64099	36450	64099	36450	64099

478	36450	72464	36450	64099	36450	64099	36450	64099
479	36450	72464	36450	64099	36450	64099	36450	64099
480	36450	72464	36450	64099	36450	64099	36450	64099
481	34992	81199	32322	73642	33558	71476	36227	65436
482	34992	81199	32322	73642	33558	71476	36117	66049
483	34992	81199	34224	73192	33336	72877	36008	66667
484	34992	81199	34224	73192	33336	72877	35955	66860
485	34992	81199	34224	73192	34216	72792	35848	67550
486	34992	81199	34224	73192	34216	72792	35796	67839
487	34992	81199	34224	73192	34992	72343	35745	68013
488	34992	81199	34224	73192	34992	72343	35721	68104
489	34992	81199	34224	73192	34992	72445	35618	68814
490	34992	81199	34224	73192	34992	72445	35568	69173
491	34992	81199	34224	73192	34992	72445	35519	69477
492	34992	81199	34224	73192	34992	72445	35496	69511
493	34992	81199	34224	73192	34992	72466	35449	69827
494	34992	81199	34224	73192	34992	72466	35427	69928
495	34992	81199	34224	73192	34992	72473	35397	69968
496	34992	81199	34224	73192	34992	72473	35397	69968
497	34992	81199	34224	73192	34992	72582	35302	70737
498	34992	81199	34224	73192	34992	72582	35256	71064
499	34992	81199	34224	73192	34992	72589	35211	71396
500	34992	81199	34224	73192	34992	72589	35190	71442
501	34992	81199	34224	73192	34992	72596	35147	71848
502	34992	81199	34224	73192	34992	72596	35127	71993
503	34992	81199	34224	73192	34992	72603	35108	72019
504	34992	81199	34224	73192	34992	72603	35100	72036
505	34992	81199	34224	73192	34992	72712	35061	72342
506	34992	81199	34224	73192	34992	72712	35043	72499
507	34992	81199	34224	73192	34992	72719	35019	72527
508	34992	81199	34224	73192	34992	72719	35019	72527
509	34992	81199	34224	73192	34992	72726	35000	72699
510	34992	81199	34224	73192	34992	72726	34992	72726
511	34992	81199	34224	73192	34224	73192	34991	72728
512	34992	81199	34224	73192	34224	73192	34224	73192
513	37908	102100	32640	79230	33096	78606	35313	74481
514	37908	102100	32640	79230	33096	78606	34739	75678
515	37908	102100	32328	81782	33012	80846	35638	76179
516	37908	102100	32328	81782	33012	80846	35721	76465
517	37908	102100	32304	83966	33216	82718	35886	77423
518	37908	102100	32304	83966	33216	82718	35290	78136
519	37908	102100	32292	85058	33318	83654	36055	78138
520	37908	102100	32292	85058	33318	83654	36099	78290
521	37908	102100	32556	86864	33810	85148	36268	79208
522	37908	102100	32556	86864	33810	85148	35726	79963
523	37908	102100	32688	87775	34056	85903	36441	80075
524	37908	102100	32688	87775	34056	85903	36486	80151
525	37908	102100	32820	88702	34302	86674	36575	80567
526	37908	102100	32820	88702	34302	86674	36621	80714
527	37908	102100	32886	89132	34425	87026	36693	80815
528	37908	102100	32886	89132	34425	87026	36693	80815
529	37908	102100	37908	86162	37908	86028	36870	81904
530	37908	102100	37908	86162	37908	86028	36960	82387
531	37908	102100	37908	86162	37908	86095	37051	82875
532	37908	102100	37908	86162	37908	86095	37098	82991
533	37908	102100	37908	86162	37908	86095	37191	83557
534	37908	102100	37908	86162	37908	86095	37239	83778
535	37908	102100	37908	86162	37908	86095	37288	83872
536	37908	102100	37908	86162	37908	86162	37314	83923
537	37908	102100	37908	86162	37908	86162	37411	84381
538	37908	102100	37908	86162	37908	86162	37461	84614
539	37908	102100	37908	86162	37908	86162	37539	84742
540	37908	102100	37908	86162	37908	86162	37539	84742
541	39366	104744	39366	90918	44550	85799	37622	85022
542	39366	104744	39366	90918	44550	85799	37665	85099
543	39366	104744	39366	90918	44550	85799	37664	85101
544	39366	104744	39366	90918	44550	85799	37665	85106
545	39366	104744	39366	90918	44550	85799	37858	86006
546	39366	104744	39366	90918	44550	85799	37956	86460
547	39366	104744	39366	90918	44550	85942	38055	86852
548	39366	104744	39366	90918	44550	85942	38106	86916
549	39366	104744	39366	90918	44550	85942	38207	87454
550	39366	104744	39366	90918	44550	85942	38259	87659
551	39366	104744	39366	90918	44550	85942	38312	87733
552	39366	104744	39366	90918	44550	85942	38340	87774
553	39366	104744	39366	90918	44550	85942	38445	88268
554	39366	104744	39366	90918	44550	86085	38499	88519
555	39366	104744	39366	90918	44550	86085	38583	88667

556	39366	104744	39366	90918	44550	86085	38583	88667
557	39366	104744	39366	90918	44550	86085	38640	88867
558	39366	104744	39366	90918	44550	86085	38718	88827
559	39366	104744	39366	90918	44550	86085	38718	88827
560	39366	104744	39366	90918	44550	86085	38718	88827
561	39366	104744	39366	90918	39366	90767	38831	89500
562	39366	104744	39366	90918	39366	90767	38889	89771
563	39366	104744	39366	90918	39366	90774	38948	90047
564	39366	104744	39366	90918	39366	90774	38979	90049
565	39366	104744	39366	90918	39366	90781	39074	90383
566	39366	104744	39366	90918	39366	90781	39123	90484
567	39366	104744	39366	90918	39366	90788	39122	90486
568	39366	104744	39366	90918	39366	90788	39123	90491
569	39366	104744	39366	90918	39366	90897	39188	90733
570	39366	104744	39366	90918	39366	90897	39276	90810
571	39366	104744	39366	90918	39366	90904	39276	90810
572	39366	104744	39366	90918	39366	90904	39276	90810
573	39366	104744	39366	90918	39366	90911	39335	90922
574	39366	104744	39366	90918	39366	90911	39366	90911
575	39366	104744	39366	90918	39366	90918	39365	90913
576	39366	104744	39366	90918	39366	90918	39366	90918
577	52650	109000	40470	99326	42546	97118	40647	92058
578	52650	109000	40470	99326	42546	97118	41289	92632
579	52650	109000	52650	94727	44208	98887	41932	93140
580	52650	109000	52650	94727	44208	98887	42255	93254
581	52650	109000	52650	94727	52650	94526	42900	93916
582	52650	109000	52650	94727	52650	94526	43224	94179
583	52650	109000	52650	94727	52650	94526	43549	94303
584	52650	109000	52650	94727	52650	94526	43713	94369
585	52650	109000	52650	94727	52650	94526	44362	94983
586	52650	109000	52650	94727	52650	94593	44688	95294
587	52650	109000	52650	94727	52650	94593	45015	95538
588	52650	109000	52650	94727	52650	94593	45180	95518
589	52650	109000	52650	94727	52650	94593	45509	95774
590	52650	109000	52650	94727	52650	94593	45675	95833
591	52650	109000	52650	94727	52650	94660	45927	95786
592	52650	109000	52650	94727	52650	94660	45927	95786
593	52650	109000	52650	94727	52650	94660	46584	96587
594	52650	109000	52650	94727	52650	94660	46914	96918
595	52650	109000	52650	94727	52650	94660	47245	97254
596	52650	109000	52650	94727	52650	94727	47412	97278
597	52650	109000	52650	94727	52650	94727	52650	94727
598	52650	109000	52650	94727	52650	94727	52650	94727
599	52650	109000	52650	94727	52650	94727	52650	94727
600	52650	109000	52650	94727	52650	94727	52650	94727

Abkürzungsverzeichnis

AES	Advanced Encryption Standard
AIA	Almost Inverse Algorithm
ASIC	Application-Specific Integrated Circuit
CA	Complexity of Algorithms
EC	Elliptic Curve
ECC	Elliptic Curve Cryptography
ECDLP	Elliptic Curve Discrete Logarithm Problem
EEA	Extended Euclidean Algorithm
FF	Flip-Flops
FPGA	Field Programmable Gate Array
FT	Fourier-Transformation
GAM	Gate Area Matrix
GC	Gate Complexity
GCM	Gate Complexity Matrix
GEM	Gate Energie consumption Matrix
GF	Galois field
GR	Große Raute
LP	Longest Path
MAIA	Modified AIA
MF	Multiplikations-Formel
MM	Multiplikations-Methode
NIST	National Institute of Standards and Technology, USA
OEF	Optimal Extended Fields
PM	Polynom-Multiplizierer
PP	Polynom-Produkt
RSA	Rivest, Shamir, Adleman, asymmetrisches Kryptosystem
TD	Tabellarische Darstellung
TM	Teil-Multiplizierer
TP	Teil-Produkt
TSP	Traveling Salesman Problem

Abbildungsverzeichnis

Abbildungen aus Kapitel 1 – Kapitel 7

Abbildungen aus Anhang 1 – Anhang 4

Tabellenverzeichnis

Tabellen aus Kapitel 1 – Kapitel 7

Tabellen aus Anhang 1 – Anhang 4